UNIVERSIDAD
POLITECNICA
DE VALENCIA

TESIS DOCTORAL

Identificación Robusta de Sistemas no Lineales mediante Algoritmos Evolutivos

Realizada por:

Juan Manuel Herrero Durá

Dirigida por:

Dr. Miguel Martínez Iranzo

Dr. Xavier Blasco Ferragud

Valencia, 2 de Octubre de 2006

TESIS DOCTORAL

Identificación Robusta de Sistemas no Lineales mediante Algoritmos Evolutivos

---

TRIBUNAL CALIFICADOR

Presidente: **Dr. D. Pedro ALBERTOS PÉREZ**
Catedrático de Universidad
Área de Ingeniería de Sistemas y Automática

Vocales: **Dr. D. Eduardo FERNÁNDEZ CAMACHO**
Catedrático de Universidad
Área de Ingeniería de Sistemas y Automática

Dr. D. César DE PRADA MORAGA
Catedrático de Universidad
Área de Ingeniería de Sistemas y Automática

Dr. D. Manuel BERENGUEL SORIA
Titular de Universidad
Área de Ingeniería de Sistemas y Automática

Secretario: **Dr. D. Javier SANCHIS SÁEZ**
Titular de Escuela Universitaria
Área de Ingeniería de Sistemas y Automática

Suplentes: **Dr. D. Manuel PÉREZ POLO**
Catedrático de Universidad
Área de Ingeniería de Sistemas y Automática

Dr. D. Rafael SANZ DOMÍNGUEZ
Catedrático de Universidad
Área de Ingeniería de Sistemas y Automática

Valencia, 2 de Octubre de 2006

Para mis Padres
mi Hermano
y Esther

AGRADECIMIENTOS

Me gustaría hacer llegar mis agradecimientos a todas las personas que, de modo alguno, me han animado y apoyado durante el desarrollo de esta Tesis Doctoral.

En primer lugar, a mis directores Miguel y Xavi por sus inestimables consejos e infinidad de horas que han dedicado a la dirección de la tesis. Al resto de compañeros del grupo del investigación de Control Predictivo: César, Javi, Jose Vte. y Serafín por el apoyo e interés que han mostrado. A los compañeros del Departamento de Ingeniería de Sistemas y Automática y en especial a la "colla" del almuerzo. Por último, con todo mi cariño, quiero dedicar este trabajo a mis amigos y familia, en especial a mis abuelos.

Resumen

Al proceso de identificación de los parámetros de un modelo nominal y su incertidumbre para su utilización en Control Robusto se le conoce como Identificación Robusta Paramétrica (IR).

Un posible enfoque para abordar la IR, que resulta apropiado cuando el desconocimiento de las propiedades estadísticas del ruido y/o la dinámica no modelada invalidan los enfoques estocásticos, es el determinístico (*Set Membership Estimation*). Este enfoque asume que el error de identificación (EI), diferencia entre las salidas medidas del proceso y las simuladas del modelo, aunque es desconocido, está acotado. De ahí que, bajo este enfoque, se persiga la determinación del conjunto de parámetros que consiguen mantener el EI acotado para una determinada norma y cota. Dicho conjunto es conocido como el conjunto de parámetros factibles (FPS).

Cuando el modelo es lineal respecto de sus parámetros, el FPS, si existe, es un politopo convexo. En modelos no lineales dicho politopo puede ser no convexo e incluso inconexo.

En esta tesis se presenta una metodología de IR que permite determinar FPS, de cualquier tipo, en modelos no lineales cualesquiera, acotando el EI simultáneamente mediante varias normas. La metodología transforma el problema de IR en un problema de optimización multimodal con infinitos óptimos globales, los cuales constituyen el FPS. Para su optimización se ha desarrollado un algoritmo evolutivo (EA) específico ϵ-GA, que caracteriza el FPS mediante un conjunto discreto de modelos FPS^* adecuadamente distribuido a lo largo del FPS.

La metodología viene acompañada de un procedimiento que facilita la determinación de las cotas, asociadas a las normas que acotan el EI, para asegurar que $FPS \neq \emptyset$. Para ello, se utiliza la información que genera el frente de Pareto resultante de la minimización simultánea de las normas mediante una optimización multiobjetivo. Para resolver este problema de optimización se ha desarrollado el algoritmo evolutivo $\not\epsilon$-MOGA.

Adicionalmente, se propone como modelo nominal un modelo de proyección interpolada restringida que, perteneciendo al FPS, resulta óptimo respecto del error de identificación y respecto del error de estimación en el espacio de parámetros.

Como ejemplos de aplicación de la metodología propuesta se presenta la IR, con datos reales, de los parámetros de tres modelos no lineales: un sistema térmico, un modelo que refleja el bloqueo que produce un determinado fármaco sobre las corrientes iónicas de una célula cardíaca y el modelo climático de un invernadero (temperatura y humedad) con cultivo hidropónico de rosas.

Resum

Al procés d'identificació dels paràmetres d'un model nominal i la seua incertesa per a la seua utilització en Control Robust se'l coneix com a Identificació Robusta Paramètrica (IR).

Un possible enfocament per a abordar l'IR, que resulta apropiat quan el desconeixement de les propietats estadístiques del soroll i/o la dinàmica no modelada invaliden els enfocaments estocàstics, és el determinístic (*Set Membership Estimation*). Aquest enfocament assumeix que l'error d'identificació (EI), diferència entre les eixides mesurades del procés i les simulades del model, encara que és desconegut, està acotat. Davall aquest enfocament, es persegueix la determinació del conjunt de paràmetres que aconsegueixen mantenir l'EI acotat per a una determinada norma i cota. Dit conjunt és conegut com el conjunt de paràmetres factibles (FPS).

Quan el model és lineal respecte dels seus paràmetres, el FPS, si existeix, és un politop convex. En models no lineals dit politop pot ser no convex i fins i tot inconnex.

En aquesta tesi es presenta una metodologia d'IR que permet determinar FPS, de qualsevol tipus, en models no lineals qualsevol, acotant l'EI simultàniament mitjançant diverses normes. La metodologia transforma el problema d'IR en un problema d'optimització multimodal amb infinits òptims globals, els quals constitueixen el FPS. Per a la seua optimització s'ha desenvolupat un algoritme evolutiu (EA) específic ϵ-GA, que caracteritza el FPS mitjançant un conjunt discret de models FPS^* adequadament distribuït al llarg del FPS.

La metodologia ve acompanyada d'un procediment que facilita la determinació de les cotes, associades a les normes que acoten l'EI, per a assegurar que $FPS \neq \emptyset$. Per a això, s'utilitza la informació que genera el front de Pareto resultant de la minimització simultània de les normes mitjançant una optimització multiobjetiu. Per a resoldre, el problema multiobjectiu s'ha desenvolupat l'algoritme evolutiu $\not\epsilon$-MOGA.

Addicionalment, es proposa com a model nominal un model de projecció interpolada restringida que, pertanyent al FPS, resulta òptim respecte de l'error d'identificació i respecte de l'error de estimació en l'espai de paràmetres.

Com a exemples d'aplicació de la metodologia proposada es presenta l'IR, amb dades reals, dels paràmetres de tres models no lineals: un sistema tèrmic, un model que reflecteix el bloqueig que produeix un determinat fàrmac sobre els corrents iònics d'una cèl·lula cardíaca i el model climàtic d'un hivernacle (temperatura i humitat) amb cultiu hidropònic de roses.

Abstract

The identification process of the parameters of a nominal model and its uncertainty, when it is used for Robust Control, is known as Parametric Robust Identification (RI).

A possible approach to RI, which is appropriate when noise statistical properties unknown and/or model error invalidate statistical approaches, is the deterministic one (*Set Membership Estimation*). This deterministic approach assumes that identification error (IE), differences between the simulated outputs of the model and the measured outputs of the process, although unknown, will be bounded. Therefore, the objective is to estimate the parameters set of a model which keeps the identification error bounded by a certain norm and bound. This set is known as the feasible parameter set (FPS).

For linear in their parameters models, the FPS is, if it exists, a convex polytope. In nonlinear models, the polytope can be non-convex even disjoint.

In this thesis a RI methodology, which permits to estimate any kind of FPS in nonlinear models when IE is bounded by several norms simultaneously, is presented. This methodology converts the RI problem into a multimodal optimization problem with optimal global infinities, which constitute the FPS. For its optimization a specific evolutionary algorithm ϵ-GA has been developed, to characterize the FPS by means of a discrete set of models FPS^* adequately distributed along the FPS.

The methodology comes accompanied by a procedure that makes easy the determination of bounds, associated to the norms of the IE, in order to guarantee an $FPS \neq \emptyset$. For that, the Pareto Front information, which is obtained by means of minimization norms of the IE in a multobjective context is used. To solve the multobjective problem an evolutionary algorithm $\not{\epsilon}$-MOGA has been developed.

In addition, a nominal model of restricted interpolated projection which belongs to the FPS is proposed. It is optimal in both identification and estimation errors in the parameter space.

The RI of three nonlinear models, with real data, is presented as application examples of the proposed methodology: a thermal process, a model which shows the blockage that produces a given drug on the ionic currents of a cardiac cell and a greenhouse climate model (temperature and humidity) with roses hydroponic crop.

Contenidos

Lista de Acrónimos

DG	Distancia generacional
EA	Algoritmo evolutivo
EC	Computación evolutiva
EI	Error de identificación
EP	Programación evolutiva
ES	Estrategía evolutiva
FP	Frente de Pareto
FPS	Conjunto de parámetros factibles
GA	Algoritmo genético
GP	Programación genética
HR	Ratio de hiper-área
IR	Identificación robusta
MO	Optimización multiobjectivo
MOEA	Algoritmo evolutivo multiobjetivo
MOGA	Algoritmo genético multiobjetivo
MOGP	Programación genética multiobjetivo
MOP	Problema de optimización multiobjetivo
MUS	Conjunto de incertidumbre en las salidas
NIND	Número de individuos no dominados
NPGA	Niched Pareto Genetic Algorithm (nombre propio)
NSGA	Nondominated Sorting Genetic Algorithm (nombre propio)
OP	Problema de optimización
PA	Potencial de acción
PAES	Pareto Archived Evolution Strategy (nombre propio)
PESA	Pareto Envelope-based Selection Algorithm (nombre propio)
PI	Regulador proporcional-integral
PID	Regulador proporcional-integral-derivativo
RBOX	Ratio box del frente de Pareto
RE	Ratio de error
SBX	Cruce por simulación binaria
SP	Espaciado del frente de Pareto
SPEA	Strength Pareto Evolutionary Algorithm (nombre propio)
SQP	Programación cuadrática secuencial
VEGA	Vector Evaluated Genetic Algorithm (nombre propio)
$\mathcal{C}$	Alcance comparativo

Capítulo 1

Motivación y Objetivos de la Tesis

1.1. Introducción

A lo largo de este capítulo se presentan las ideas que motivan el desarrollo de la tesis la cual se enmarca dentro del campo de la identificación robusta no lineal, así como los objetivos que se pretenden alcanzar con la misma. En la última parte del capítulo, se presenta la organización de la tesis mostrando, de forma breve, el contenido de los capítulos que la componen.

1.2. Motivación

Uno de los primeros pasos en muchas áreas tecnológicas es la obtención de un modelo matemático (modelado) que describa el comportamiento de un determinado sistema o proceso. En control de procesos, por ejemplo, el modelado es un aspecto muy importante que, en parte, determinará la calidad del comportamiento final del bucle de control. Para que el modelo pueda ser aplicado con éxito, debe ajustarse al proceso en cuestión, no sólo en términos de estructura sino también a nivel de sus parámetros. Por ello, es necesaria la identificación de éstos a través de la información propia del proceso, la cual puede ser obtenida mediante experimentos (observaciones de las entradas y salidas del proceso).

Durante los últimos años ha aumentado considerablemente el interés por incorporar, de forma explícita, el efecto de las posibles no linealidades del proceso sobre la estructura del modelo construido utilizando primeros principios. Cuando se adopta este enfoque, modelar la dinámica a través de primeros principios, el problema recae, en última instancia, en la identificación de los parámetros del modelo.

El hecho de que no se conozca por completo el comportamiento del proceso y que los datos disponibles pueden ser escasos y/o poco fiables, dificulta el proceso de identificación. Los parámetros, identificados en estas condiciones, presentarán incertidumbre (debido a la dinámica no modela y/o el ruido de medida) que debería tenerse en cuenta cuando se utilice el modelo para realizar predicciones, diseñar el controlador, etc. Al proceso de identificación del modelo nominal y su incertidumbre, para el control robusto (ya sea como punto de partida para el diseño del controlador robusto o como mecanismo para establecer las predicciones) se le conoce como identificación robusta o identificación robusta paramétrica (IR).

A la hora de abordar la IR es posible establecer dos enfoques diferentes: estocástico o determinístico. El primero de ellos asume que el error de identificación (diferencia entre las salidas observadas del proceso y las simuladas del modelo) puede ser modelado como una variable aleatoria con ciertas propiedades estadísticas. Bajo este enfoque es posible utilizar las técnicas clásicas de identificación [108, 102] para determinar el modelo nominal, óptimo respecto de un determinado criterio, y su incertidumbre que vendrá asociada a la información de la matriz de covarianza de los parámetros estimados. Cuando dichas asunciones no resultan adecuadas (tratar los errores como variables aleatorias exclusivamente) o cuando existen errores de modelado inevitables (situación común en el modelado de pro-

cesos físicos) resulta más apropiado aplicar el enfoque determinístico [131, 159, 126]. En este segundo enfoque se asume que el error de identificación (EI), aunque es desconocido, está acotado, lo que puede ser más realista en ciertos casos.

El objetivo del enfoque determinístico será la obtención del modelo nominal y su incertidumbre o directamente el conjunto de parámetros factibles (FPS), es decir, el conjunto de parámetros que consiguen mantener el EI acotado, para una determinada función o norma del EI y su cota. Por lo tanto, el FPS vendrá condicionado por la norma que se utilicen para acotar el EI y en especial por su correspondiente cota[1].

A la hora de establecer dicha cota se debe utilizar la información que se tenga *a priori* del proceso (p.e. dinámica no modelada) y las características del ruido. Sin embargo, dada la dificultad que esto puede entrañar, en muchos casos, se termina eligiendo la cota atendiendo a las prestaciones que se desean para las predicciones del modelo. Si la cota es muy baja el FPS podría terminar siendo un conjunto vacío, mientras que una cota excesivamente alta conllevaría un FPS conservativo (más grande de lo necesario), de ahí que, el proceso de determinación de la cota del EI resulte especialmente crítico.

Cuando el modelo del proceso es lineal respecto de sus parámetros, el FPS, si existe, es un politopo convexo más o menos complicado. Sin embargo, cuando el modelo es no lineal, el FPS puede resultar un politopo no convexo y/o inconexo dificultándose su determinación, de ahí que suela ser aproximado por ortotopos, elipsoides, paralelotopos, etc. lo que conlleva, en general, a una determinación del FPS más conservativa.

Existen técnicas como el análisis intervalar, las máquinas de vector soporte y otras que aunque no aproximan al FPS presentan limitaciones (en cuanto a la forma del FPS o el tipo de función que se puede utilizar para acotar el EI) o su utilización resulta muy complicada cuando el modelo es complejo (no diferenciable respecto del sus parámetros, discontinuidades en parámetros y/o señales, etc.).

1.3. Objetivos y solución adoptada

Como consecuencia de las deficiencias que presentan las herramientas actuales de IR, descritas en la sección anterior, se plantea como objetivo principal de la tesis el desarrollo de una metodología para la caracterización del FPS que presente las siguientes características:

1. Sea flexible, de manera que pueda:
   - Utilizar cualquier tipo de modelo para el proceso.
   - Acotar el error de identificación mediante diferentes normas simultáneamente sin restricciones sobre el tipo de norma utilizada. De esta manera se plantea la posibilidad de tener en cuenta diferentes consideraciones al mismo tiempo,

[1]Por ejemplo, el FPS consistente con la norma-∞ y cota η correspondería al conjunto de parámetros que consiguen que el EI nunca fuese mayor a η en ninguna de las observaciones de la salida.

por ejemplo, acotar el error a nivel de muestra y su integral simultáneamente, tratar el error de identificación de cada salida de forma independiente sin tener que escalarlas, etc.

- Caracterizar FPS de cualquier tipo: convexos, no convexos e inconexos.

2. Ser no conservativa, es decir, la solución no debe ser una aproximación del FPS mediante hipervolúmenes estándar y a ser posible facilitar la determinación de las cotas asociadas a las diferentes normas con la intención de asegurar que $FPS \neq \emptyset$.

3. Ser eficiente desde el punto de vista computacional.

La metodología planteada en esta tesis para cumplir dichas características se basará en la optimización de una función, que se construirá a partir de las normas a utilizar sobre el EI y sus cotas, cuyos mínimos globales constituirán el FPS. Por lo tanto, será una función multimodal que además podrá ser no convexa y/o presentar mínimos locales por lo que los optimizadores tradicionales (por ejemplo, tipo SQP) resultarían inadecuados.

Para escoger las cotas y evitar un $FPS = \emptyset$ se propondrá un procedimiento que utilizará la información del frente de Pareto que se obtiene de la minimización de las normas sobre el EI de forma simultánea, a través de una optimización multiobjetivo (MO). Dicho problema MO podría ser no convexo, presentar mínimos locales y/o discontinuidades en el frente de Pareto lo que dificultaría su determinación.

A raíz de la solución planteada en esta tesis aparece la necesidad de abordar dos problemas de optimización:

- Una optimización de una función multimodal.
- Una optimización multiobjetivo.

Para ello, se utilizarán dos algoritmos evolutivos (EAs). Los EAs son herramientas de optimización global robustas y muy flexibles ya que pueden tratar con éxito una gran variedad de problemas de diferentes áreas, donde otros métodos encuentran dificultades, de ahí que sean una alternativa cada vez más empleada. Los EAs también han demostrado su gran potencial a la hora de encontrar la solución al problema MO, ante funciones de cualquier índole, obteniendo en una única ejecución el conjunto de soluciones al problema MO.

Por lo tanto serán objetivos adicionales de la tesis:

1. El estudio de los EAs en todos sus aspectos: optimización de funciones multimodales, optimización multiobjetivo, etc.

2. El desarrollado de un algoritmo evolutivo específico para optimizar funciones multimodales con infinitos óptimos globales.

3. El desarrollado de un algoritmo evolutivo multiobjetivo que genere frentes de Pareto bien distribuidos.

Como aplicaciones reales de la metodología se planteará como objetivo la identificación robusta de:

- Un modelo sencillo para un sistema térmico donde se modelan los fenómenos de conducción y convección térmica y que es no lineal respecto de sus parámetros y una de sus entradas.

- Un modelo biomédico que refleja el bloqueo que produce un fármaco sobre los canales iónicos de una célula cardíaca. Se trata de un modelo no lineal respecto de sus parámetros que presenta un nivel de ruido de medida considerable y donde la dinámica no modelada es evidente dado el alto grado de abstracción que se produce al intentar reproducir el comportamiento real de este proceso.

- Un modelo climático (temperatura y humedad) de un invernadero con cultivo hidropónico de rosas. El modelo además de modelar los balances de masa y energía más importantes incluye el comportamiento biológico de las plantas lo que hace de él un modelo con no linealidades importantes y con un número de parámetros a identificar elevado. De nuevo la presencia de ruido de medida y dinámica no modelada es evidente.

La determinación del FPS permitirá la realización de predicciones robustas que podrían ser utilizadas para analizar el efecto que ejercen los parámetros sobre el comportamiento del modelo (por ejemplo en caso del modelo biomédico) o para realizar un control robusto (por ejemplo del clima del invernadero).

1.4. Organización de la tesis

La organización de la tesis se ha establecido teniendo en cuenta que, la identificación robusta en modelos no lineales IR, se abordará mediante la utilización de un EA específico desarrollado para optimizar funciones multimodales con infinitos mínimos globales denominado ϵ-GA y otro que, permita obtener el frente de Pareto asociado a una optimización multiobjetivo llamado $\not\epsilon^2$MOGA. De ahí que inicialmente se establezcan las bases de los EAs (capítulo 2) y los MOEAs (capítulo 3) para poder entender los algoritmos $\not\epsilon^2$MOGA y ϵ-GA (capítulos 4 y 5 respectivamente). A continuación, se presentan los conceptos básicos de IR (capítulo 6) para después presentar la metodología de IR propuesta en esta tesis (capítulo 7) y su aplicación a la IR de tres procesos reales, un sistema térmico, un modelo de bloqueo de fármaco en células cardíacas y un modelo climático de un invernadero (capítulo 8). Por último, se presentan las conclusiones más relevantes de la tesis (capítulo 9).

A continuación, para que el lector tenga una visión global, se presentan los contenidos de los capítulos y apéndices de la tesis y los objetivos que se persiguen en cada uno de ellos.

- **Capítulo 2. Algoritmos Evolutivos.** Este capítulo presenta los aspectos básicos de los Algoritmos Evolutivos (EA) relacionados con el tipo de codificación a utilizar (binaria o real), operadores (selección/determinación, cruce y mutación), ajuste de parámetros del propio EA, etc. y su utilidad como herramienta de optimización global. El capítulo también se centra en aspectos particulares como la incorporación de las restricciones en el problema de optimización, implementación paralela de EAs, y de mecanismos específicos que permitan abordar la optimización de funciones multimodales.

- **Capítulo 3. Algoritmos Evolutivos Multiobjetivo.** Al inicio del capítulo se muestran los conceptos básicos y la terminología asociada a los problemas de optimización multiobjetivo (MOPs) así como los métodos clásicos de resolución de MOPs. A continuación, se presentan los EAs más importantes utilizados en la resolución de MOPs clasificándolos como de primera y segunda generación. Por último, se presentan una serie de métricas que permitirán analizar el funcionamiento de los MOEAs y compararlos.

- **Capítulo 4. Algoritmo Evolutivo $\not\epsilon$-MOGA.** En este capítulo se presenta, inicialmente, la implementación de un MOEA denominado ϵ-MOGA basado en el algoritmo ϵ-MOEA presentado en [45]. Ambos algoritmos, junto con los algoritmos de búsqueda aleatoria y exhaustiva se comparan, utilizando los problemas MOP1...MOP5 presentados en el apéndice A, mostrando que ϵ-MOGA presenta mejores prestaciones. A continuación, se presenta una variante original del algoritmo ϵ-MOGA, el algoritmo $\not\epsilon$MOGA que permite ajustar dinámicamente los límites del frente de Pareto. Por último, como ejemplo, se muestra la potencia del algoritmo resolviendo un ejemplo de ingeniería donde se plantea un problema de optimización multiobjetivo para determinar las secciones de tres barras de una estructura mecánica.

- **Capítulo 5. Algoritmo Evolutivo ϵ-GA.** En este capítulo se presenta la implementación de un EA denominado ϵ-GA diseñado específicamente para optimizar funciones mono objetivo multimodales que presentan múltiples (incluso infinitos) óptimos globales. El algoritmo presenta la misma estructura que el algoritmo multiobjetivo ϵ-MOGA en el que está inspirado. Para evaluar las prestaciones del algoritmo ϵ-GA se han desarrollado un conjunto de funciones multimodales de test (OP1 ...OP5) que se presentan en el apéndice A y se ha comparado los resultados con los obtenidos por dos estrategias basadas en el algoritmo SQP (*multi-start* SQP).

- **Capítulo 6. Identificación Paramétrica de Sistemas.** Este capítulo presenta, a modo de resumen, los aspectos más importantes relacionados con la identificación paramétrica de sistemas, centrando el interés en el uso de modelos no lineales respecto de sus parámetros obtenidos a partir de primeros principios y la necesidad de

utilizar optimizadores globales (por ejemplo, EAs) para abordar los problemas de optimización (no convexos, multimodales, etc.) que pueden resultar cuando se utilizan modelos no lineales y/o criterios de optimalidad cualesquiera. A continuación, se presenta una breve referencia bibliográfica de trabajos donde se utilizan EAs para la identificación de modelos no lineales con criterios de optimalidad no cuadráticos. Para terminar el capítulo se dedica una sección a presentar los conceptos y formulación asociada a la identificación robusta cuando se utiliza un enfoque determinístico. Tras definir el FPS (solución al problema de IR) se muestran diferentes métodos para su determinación cuando se utilizan modelos no lineales (FPS podría ser no convexo y/o inconexo dificultando su obtención) analizando sus limitaciones.

- **Capítulo 7. Identificación Robusta de Sistemas No Lineales mediante ϵ-GA y $\not\epsilon$-MOGA.** En este capítulo se produce la sinergia de los anteriores para mostrar la metodología flexible y computacionalmente eficiente, basada en los algoritmos $\not\epsilon$-MOGA y ϵ-GA que permite caracterizar el conjunto de parámetros factibles FPS cuando se plantea un problema de identificación robusta paramétrica en modelos no lineales. Inicialmente, se describe la metodología desarrollada la cual pasa por crear una función a optimizar, cuyos mínimos constituyen el FPS, cuando se acota el EI por varias normas simultáneamente y se utiliza el algoritmo ϵ-GA para dar con dichos mínimos. Para ayudar a determinar la cotas del EI se propone un procedimiento basado en la información que genera el frente de Pareto que se obtiene de la minimización de las diferentes normas, para lo cual se utiliza el algoritmo $\not\epsilon$-MOGA. La determinación del FPS y el del frente de Pareto posibilita la determinación del modelo de proyección interpolada restringida propuesto en la tesis. A continuación, se establecen los criterios que se utilizarán en la validación del FPS identificado y del modelo nominal. Para terminar, se presentan dos ejemplos de IR con datos reales de un sistema térmico y del bloqueo que produce un fármaco sobre una célula cardíaca, que sirven para mostrar la metodología de IR propuesta.

- **Capítulo 8. Identificación Robusta del Modelo Climático de un Invernadero.** Tras una breve introducción, donde se pone de manifiesto la importancia de disponer de modelos que permitan aplicar estrategias de control para mejorar tanto la calidad como la cantidad de la producción, se presenta el modelado climático (temperatura y humedad) del invernadero. Tras esto, se aborda la identificación robusta del modelo climático aplicando la metodología presentada en el capítulo anterior.

- **Capítulo 9. Conclusiones y Líneas Futuras.** Este capítulo presenta las conclusiones más importantes de la tesis destacando las aportaciones, en cuanto a la metodología de IR propuesta y algoritmos desarrollados se refiere. También se presentan las líneas futuras de la tesis encaminadas a: mejorar los algoritmos desarrollados, aplicar la metodología de IR sobre otros procesos y enlazar con el diseño del control robusto a partir del FPS determinado.

- **Apéndice A. Problemas de Optimización.** En este apéndice se muestran los problemas de optimización utilizados para evaluar los algoritmos $\not\epsilon$-MOGA (proble-

mas MOP1 a MOP5 que se caracterizan por contener funciones no lineales, frentes de Pareto no convexos, discontinuos y no uniformes) y ϵ-GA (problemas OP1 a OP5 que se caracterizan expresamente por contener funciones a optimizar que son no lineales y presentan infinitos óptimos globales y además óptimos locales).

- **Apéndice B. Modelo del Invernadero y Datos para la Identificación.** Este apéndice recoge los detalles de las ecuaciones que completan el modelo climático del invernadero presentado en el capítulo 8, así como las variables y parámetros de éste, en particular, los rangos de los parámetros inciertos. También, se incluyen los datos utilizados para la identificación y validación del modelo.

Capítulo 2

Algoritmos Evolutivos

2.1. Introducción

Los **Algoritmos Evolutivos** (EA) [8] también conocidos como Computación Evolutiva (EC) son técnicas de optimización estocásticas inspiradas en la teoría evolutiva NeoDarwiniana. Esta teoría explica el proceso evolutivo de la especies, mediante el cual, aquellos individuos de la población que están mejor adaptados a su entorno, tienen mayores posibilidades de sobrevivir.

Los EAs utilizan modelos computacionales para "reproducir" los procesos evolutivos y de herencia de las especies, dentro de un proceso computacional de optimización y búsqueda. Para ello, utilizan una población de individuos a la que hacen evolucionar mediante reglas de selección, operadores de recombinación (cruce) y mutación dentro de un proceso iterativo o generacional.

Fue a finales de 1950 y principios de 1960 (ver revisión histórica en [31, 54] y sus referencias) cuando los modelos computacionales de la genética poblacional se volvieron populares forjando los orígenes de lo que ahora entendemos por Algoritmos Evolutivos. En esta época los trabajos de Alexander Fraser relacionados con la evolución de los sistemas biológicos en un ordenador, los de R.M. Friedberg intentando evolucionar programas de ordenador y los de N.A. Barricelli con las primeras simulaciones de un sistema evolutivo en un ordenador, vaticinarían, lo que más tarde serían los primeros Algoritmos Evolutivos simples. Sería H.J. Bremermann el primero en ver la evolución como un proceso de optimización, planteando una técnica evolutiva para resolver problemas de optimización con restricciones y siendo uno de los primeros en utilizar el concepto de población.

Los tres principales paradigmas de los Algoritmos Evolutivos estaban aún por llegar. Sería a mediados de 1960 cuando Fogel Owens y Walsh desarrollarían la **Programación Evolutiva** (EP) [56] una técnica que consistía en hacer evolucionar una población de autómatas de estados finitos usando un operador de mutación para modificar las transiciones y estados de los autómatas sin utilizar el operador de recombinación.

Prácticamente al mismo tiempo, en Alemania, I. Rechenberg desarrollaba un método de ajustes aleatorios inspirado en los mecanismos de la mutación dando lugar al concepto de **Estrategia Evolutiva** (ES) [138] concebido principalmente como un optimizador de parámetros reales. Junto con Rechenberg, H. Schwefel (se encargó de la implementación de una ES en un ordenador) sería quien con sus publicaciones atraería la atención de los investigadores sobre esta técnica.

Los **Algoritmos Genéticos** (GA) de John Holland [83, 63, 120] son la técnica, de entre los Algoritmos Evolutivos, más popular en la actualidad. Fueron concebidos como un algoritmo para resolver problemas de muy diversa naturaleza, lo que posiblemente haya potenciado su uso. Los GAs constituyen el paradigma más completo de la computación evolutiva cubriendo todas las ideas fundamentales de los mecanismos de la evolución natural (selección y reproducción a través del cruce y la mutación). Una rama de los GAs, la **Programación Genética** (GP) [99] basada en la evolución de los programas de ordenador usando una representación de árbol fue desarrollada hacía finales de 1980. Esta técnica es casi independiente del dominio y ha sido utilizada en muchos campos diferentes

de compresión de imágenes, diseño de circuitos, etc.

Las investigaciones en EAs no han cesado durante todo este tiempo dando pie a múltiples modificaciones de sus algoritmos progenitores, hasta el punto que cada vez es más difícil distinguir las diferencias entre los distintos tipos de los EAs existentes.

El poder de los EAs reside en el hecho de que se trata de una técnica robusta y muy flexible, ya que puede tratar con éxito una gran variedad de problemas de diferentes áreas donde otros métodos encuentran dificultades. Los EAs no garantizan la localización de la solución óptima del problema, sin embargo, existe evidencia empírica de que encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria.

Si existen técnicas específicas para resolver un determinado problema, lo más probable es que superen a los EAs, tanto en rapidez como en eficacia, de ahí que el campo de aplicación de los EAs esté especialmente relacionado con aquellos problemas para los que, por su dificultad, no existen técnicas especializadas.

De forma genérica algunas de las ventajas de los EAs son:

- Amplia aplicabilidad sin necesitar un conocimiento específico del problema a resolver.
- Simplicidad conceptual.
- Pueden incorporar conocimiento del dominio e hibridarse con otras técnicas de optimización.
- Pueden paralelizarse fácilmente.
- Robustos y poco sensibles a ser atrapados en óptimos locales gracias a su trabajo con una población y no con un sólo individuo.

No obstante, los EAs han sido muy criticados porque se creía que funcionaban como una simple búsqueda aleatoria o, como, un algoritmo de escalado de colina que empieza en varios puntos. Sin embargo, los resultados obtenidos en los últimos años en multitud de áreas diferentes han puesto de manifiesto la inexactitud de esta apreciación.

El resto del capítulo está organizado de la siguiente manera: la sección 2.2 suministra una descripción de un EA simple mostrando las principales diferencias entre los ES, EP y GA y, plantea el principal dominio de aplicación de los EA, es decir, la optimización de funciones. En la sección 2.3, se tratan los aspectos relacionados con la codificación de los individuos de la población, en concreto con la codificación binaria y la real. La siguiente sección trata los temas relacionados con la elección del tamaño de la población y la inicialización de la misma. La sección 2.5 presenta los métodos más utilizados en la selección de individuos de unas generaciones a otras. Los diferentes operadores genéticos (cruce y mutación) tanto para la codificación binaria como real son presentados en la sección 2.6. La sección 2.7 está referida a los aspectos relacionados con las características de las funciones a optimizar, por ejemplo la forma, costes computaciones, óptimos locales,

etc. Las diferentes estructuras de computación paralela empleadas en la implementación del EA son presentadas en la sección 2.8, mientras que los procedimientos utilizados en el ajuste de los parámetros del EA son descritos en la 2.9. Los diferentes enfoques para incorporar las restricciones a una optimización con EAs son detallados en la sección 2.10. Y para terminar, antes de las conclusiones del capítulo, se dedica la sección 2.11 a los aspectos relacionados con la determinación de los óptimos de funciones multimodales.

2.2. Descripción de los EA

Un EA simple responde, desde el punto de vista de programación, a la estructura de la figura 2.1.

```
t=0
Inicializar Población P(t)
Evaluar Población P(t)
While (not Criterio Terminación) do
    t=t+1
    Seleccionar P'(t) de P(t-1)
    Alterar P'(t)
    Evaluar Población P'(t)
    Determinar P(t) a partir de P(t-1) y P'(t)
End
```

Figura 2.1: Estructura de un Algoritmo Evolutivo simple.

El EA consta de una Población $P(t) = [p^1 \dots p^N]$ de N individuos o miembros que evolucionan, a partir de una población inicial $P(0)$ y a lo largo de las generaciones, dentro de un proceso iterativo.

Cada individuo p^i está compuesto, principalmente, por su *cromosoma* (compuesto por un número L de genes)[1] que adecuadamente codificado representa un punto dentro del espacio de búsqueda solución al problema de optimización y su *fitness* que suministra información del grado de adaptación del individuo al problema de optimización.

En la fase *Seleccionar* se genera la población $P'(t)$ a partir de individuos de la generación anterior $P(t-1)$ para la reproducción, usando su $fitness$ (previamente calculado mediante la función de $fitness$) y favoreciendo a aquellos individuos con mejor grado de adaptación. La fase *Alterar* pretende hacer evolucionar los individuos hacía regiones del espacio de búsqueda mejores (en términos de valor del *fitness*) mediante la alteración de sus genes utilizando operadores genéticos como recombinación y mutación.

[1]Se denomina *gen* a la subsección de un cromosoma que codifica un parámetro. A la codificación de todos los parámetros que representan una solución al problema de optimización se le conoce como *genotipo* y *fenotipo* a los valores obtenidos de la decodificación del *genotipo* expresados en la representación usada por la función objetivo, también conocida como función de $fitness$.

El operador recombinación promueve el intercambio de información genética entre individuos padres para producir (mediante cruce) descendientes. El operador mutación, por otra parte, se encarga de alterar la información genética de la población introduciendo cambios en algunos individuos.

A continuación se evalúa la nueva población $P'(t)$ y finalmente se determina a partir de $P(t-1)$ y $P'(t)$ los individuos que conformarán la población $P(t)$. Este proceso de Selección, Alteración (recombinación y mutación), Evaluación y Determinación de la población $P(t)$ se repite hasta que el criterio de terminación es alcanzado, pudiendo ser, por ejemplo, un número máximo de generaciones.

Las diferentes etapas representadas en la estructura del EA simple presentada en la figura 2.1 cumplen un papel diferente en función del EA que se esté implementando. Por ejemplo, si el EA a implementar es un EP, el cromosoma no es codificado y es tratado directamente a nivel de *fenotipo*, de tal manera que pueda representar adecuadamente un autómata con sus estados, transiciones (en función de las entradas) y salidas. La generación de la población inicial suele ser aleatoria y la selección se realiza de forma aleatoria. Sólo utiliza el operador de mutación y la determinación se resuelve mediante un proceso de torneo probabilístico (ver sección 2.5.2).

Si el algoritmo implementado responde al paradigma ES utiliza una representación, al igual que la EP, a nivel de *fenotipo* (generalmente números reales que se utilizan directamente en la función de *fitness*). En su versión (μ+1)-ES [2], la población $P'(t)$ la formaría un único individuo obtenido utilizando un método de selección aleatorio. Sobre este individuo se aplicaría el operador de mutación de tipo gaussiana (de media nula y desviación σ). También es posible utilizar en este caso el operador de cruce seleccionando dos individuos en vez de uno. En la fase de determinación, la población $P(t)$ estaría formada por la población los μ más aptos de entre los $\mu+1$ individuos (individuos en $P(t-1)$ y $P'(t)$). A este tipo de determinación se le conoce como *extintiva* por que los individuos peores no pueden sobrevivir en poblaciones posteriores.

En cuanto al GA se refiere, la codificación utilizada más frecuentemente es la binaria (*genotipo*) siendo necesaria su decodificación para obtener el *fenotipo* correspondiente. La fase de selección es probabilística en base a la aptitud de los individuos (valor de la función de *fitness*) y normalmente el tamaño de la población $P'(t)$ es también N. Los operadores de cruce y mutación son utilizados secuencialmente para hacer evolucionar la población. La fase de determinación no existe, simplemente $P'(t)$ pasa a ser la población $P(t)$.

En la actualidad existen EAs que añaden etapas nuevas, no recogidas en la estructura simple de la figura 2.1 o que utilizan éstas pero modificadas, dando pie a un gran abanico de posibles EAs, en ocasiones, difícilmente clasificables de acuerdo con los paradigmas EP, ES y GA. Sin embargo, todos ellas tienen en común lo siguiente:

[2] En la versión original (1+1)-ES a partir de un único padre se genera un hijo que reemplaza (si mejora sus cualidades) al padre. En la versión (μ+λ)-ES a partir de μ padres se generan λ hijos, sobreviviendo los μ mejores de entre los padres e hijos. Otra variante es la (μ,λ)-ES, donde se generan λ hijos a partir de μ padres y sobreviven los μ mejores hijos.

- Una representación adecuada de las soluciones al problema planteado.
- Creación de una población inicial.
- Una función de $fitness$ que determina el nivel de aptitud de cada individuo.
- Un proceso de selección y/o determinación.
- Unos operadores genéticos.
- Los valores de parámetros que configuran el EA.

A continuación, y antes de empezar a detallar las diferentes etapas del EA mencionadas a lo largo de esta sección, se procede a definir el problema de minimización global que será tratado a lo largo del capítulo [8].

Definición 2.1 (Mínimo Global): *Dado un dominio finito $D \neq \emptyset$ y una función a optimizar $J : D \rightarrow \mathcal{R}$, el valor $J^* := J(\theta^*) > -\infty$ se conoce como* ***mínimo global*** *de J si y sólo si*

$$\forall \theta \in D : J(\theta^*) \leq J(\theta). \tag{2.1}$$

$\triangle$

Por lo tanto,

$$\theta^* = arg \min_{\theta \in D} J(\theta) \tag{2.2}$$

será la solución global mínima[3] de J para el espacio de búsqueda D.

A partir de la propia definición se puede concluir que el problema de optimización global, de una única función, tiene solución única en el espacio de la función de *fitness*, aunque no necesariamente en el espacio de búsqueda de las soluciones, D. En los casos en los que la desigualdad sea estricta, la solución global óptimo se denomina estricta.

Otra definición interesante es la de mínimo local,

Definición 2.2 (Mínimo local): *Dado un dominio finito $D \neq \emptyset$ y una función a optimizar $J : D \rightarrow \mathcal{R}$, el valor $\hat{J} := J(\hat{\theta}) > J^*$ se conoce como* ***mínimo local*** *de J si y sólo si*

$$\exists E(\theta), entorno\ de\ \theta : \forall \theta \in E(\theta) \Rightarrow J(\hat{\theta}) \leq J(\theta). \tag{2.3}$$

$\triangle$

[3]La restricción al problema de minimización no supone pérdida de generalidad alguna, ya que, $arg\ \min_{\theta \in D} J(\theta) = arg\ \max_{\theta \in D} -J(\theta)$.

2.3. Codificación

Cada individuo diferente que puede existir en una población debe tener asociado un cromosoma diferente, al mismo tiempo, dos individuos iguales deben tener el mismo cromosoma. La codificación más adecuada para el *cromosoma* del individuo depende del problema a tratar. Una buena elección de dicha codificación puede simplificar tremendamente el proceso de búsqueda del EA. Por otra parte, la codificación condicionará, en gran medida, los operadores genéticos a utilizar. Aunque existen gran cantidad de codificaciones posibles, las más comunes son la codificación binaria y la real para dominios numéricos que son las que serán tratadas en esta tesis.

2.3.1. Codificación binaria

Es la forma más tradicional de codificar el cromosoma cuando se utiliza un GA[4]. Consiste en asignar a cada posible cromosoma una cadena binaria, es decir, una cadena de unos y ceros (por ejemplo, el cromosoma de un determinado individuo podría ser la cadena 011110001101).

El hecho de que los cromosomas sean de longitud finita, limita el número de puntos del espacio de soluciones que pueden ser codificados, por ejemplo, con un cromosoma de longitud siete, sólo disponemos de 2^7 cadenas diferentes. Por ello, se deben seleccionar los puntos del espacio a los que se asignará cada cadena. Para realizar esta discretización existen distintas posibilidades y la que se utiliza generalmente es una discretización lineal. Si se quiere codificar los puntos de un espacio de búsqueda unidimensional $[\theta_{min}, \theta_{max}[= [-5, 5[$ mediante una codificación binaria con cadenas de longitud 10 y con una discretización lineal (ver figura 2.2) se obtiene:

Valor de θ	Cromosoma S
-5	0000000000
-5+d	0000000001
-5+2d	0000000010
-5+3d	0000000011
...	...
-5+(2^{10}-1)d	1111111111

Donde la precisión, en este caso, es d=$10/2^{10}$.

Por otra parte, será necesario utilizar una función de decodificación (salvo que la función de coste se pueda evaluar a partir del código binario) que convierta el cromosoma (*genotipo*) en el valor real correspondiente (*fenotipo*) dentro del espacio de búsqueda. Por

[4]En un principio, cualquier problema de representación de parámetros puede ser codificado mediante una representación binaria.

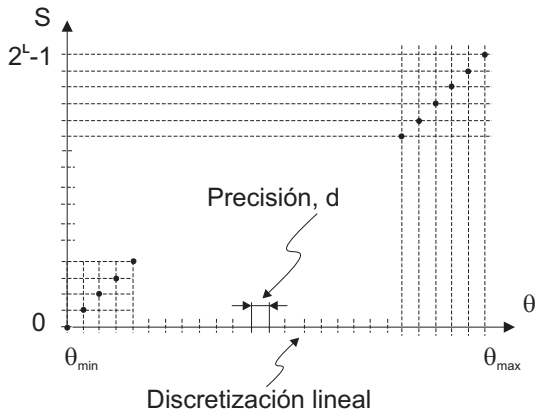

Figura 2.2: Discretización lineal de un espacio unidimensional.

ejemplo, la función

$$\theta = \theta_{min} + \frac{\theta_{max} - \theta_{min}}{2^l - 1} \left(\sum_{i=0}^{L-1} S_i \cdot 2^i \right) \tag{2.4}$$

donde L es el número de bits utilizados en la codificación y S_i representa el bit i de la codificación binaria.

Esta codificación, se puede extender al caso de espacios multidimensionales sin más que asignar una parte del cromosoma a la codificación de cada dimensión. Por ejemplo, si se quiere codificar un espacio tridimensional, $[\theta_{1min}, \theta_{1max}] \times [\theta_{2min}, \theta_{2max}] \times [\theta_{3min}, \theta_{3max}]$, con cromosomas de longitud 18 bits, se pueden asignar seis bits a cada una de las dimensiones.

$$\text{Cromosoma} \rightarrow \underbrace{100111}_{\theta_1}\underbrace{001101}_{\theta_2}\underbrace{101100}_{\theta_3}$$

Aunque la codificación más utilizada es la codificación binaria estándar también se pueden emplear otro tipo de representaciones, como por ejemplo la codificación "Gray" [28] que representa cada número con una cadena de unos y ceros (cadena binaria) pero con una propiedad particular: dos números consecutivos difieren únicamente en un bit. Existen diversas formas para esta codificación pero la que más se utiliza es la que se conoce como código Gray reflejado. En la tabla 2.1 se muestra un ejemplo comparándolo con la codificación binaria estándar.

Cuando se aplica la codificación binaria Gray en un algoritmo genético, no se aprecia un comportamiento muy diferente al de la codificación binaria estándar, los resultados que se obtienen son similares [19]. Al mismo tiempo, en ambas codificaciones aparece el problema de que pequeños cambios en el *cromosoma*, pueden producir grandes saltos en el espacio de búsqueda. Por otra parte, si se disponen de un número elevado de variables reales a codificar y se desea una buena precisión, las cadenas binarias serán extremadamente largas y el EA tendrá unas prestaciones pobres.

Número Entero	Código binario Estándar	Código binario Gray
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Tabla 2.1: Comparación de los códigos binarios estándar y Gray para la codificación de enteros del 0 al 7.

2.3.2. Codificación real

En la codificación real [120] cada parámetro de la función de *fitness* está representado por un número real, por tanto, cada cromosoma está formado por una cadena de números reales $p^i = [\theta_1^{p_i}, \ldots, \theta_L^{p_i}]$, tantos como la dimensión del espacio de búsqueda[5].

Por ejemplo, si se quiere codificar un espacio tridimensional $L = 3$ como:

$$[\theta_{1min}, \theta_{1max}] \times [\theta_{2min}, \theta_{2max}] \times [\theta_{3min}, \theta_{3max}],$$

$$\text{Cromosoma} \rightarrow \underbrace{34.1235}_{\theta_1} | \underbrace{123.0293}_{\theta_2} | \underbrace{0.1203}_{\theta_3}.$$

La característica fundamental de este tipo de codificación es que no discretiza el espacio de búsqueda[6]. Esto se puede considerar como una ventaja salvo que el espacio de búsqueda sea discreto y por lo tanto, esta codificación no se puede aplicar.

En la codificación binaria, existe un número finito de puntos en el espacio de búsqueda que depende de la longitud del cromosoma.

En el caso de la codificación real, el espacio es continuo, existen infinitos puntos en él, se pueden alcanzar soluciones con una precisión mayor sin incrementar la longitud del cromosoma, y por lo tanto, sin incrementar el coste computacional (la longitud del cromosoma afecta al coste del algoritmo en la codificación binaria). Otra ventaja de la codificación real es que la evaluación de la función de coste no necesita una operación

[5]La codificación real ha sido utilizada ampliamente en ES, EP y también en GAs. Aunque en estos últimos la codificación más utilizada es la binaria, existen bastantes aplicaciones donde se ha utilizado la codificación real. Hoy en día existen defensores y detractores de ambas codificaciones y aún no se ha llegado a un consenso sobre cual de las dos codificaciones es más efectiva.

[6]Si no se tiene en cuenta la discretización propia del ordenador que realiza los cálculos.

de decodificación para extraer los valores de los parámetros de la cadena que forma el cromosoma, ya que se utilizan directamente los componentes del cromosoma.

2.4. Tamaño e inicialización de la población

El tamaño de la población es uno de los parámetros más importantes a elegir en la mayoría de los EAs. Su elección puede resultar crítica, ya que, un número reducido de individuos podría producir una convergencia prematura[7] del algoritmo mientras que, un número elevado conllevaría un aumento considerable del coste computacional.

Son muchos los trabajos que plantean procedimientos para determinar el tamaño óptimo de la población, especialmente para GAs. Por ejemplo, Goldberg [64] determinó la siguiente expresión para fijar el número N de individuos de la población:

$$N = 1.65^{(2^{0.21 \cdot L})}, \quad L = \text{longitud cadena binaria},$$

lo que supone que el valor de N sea muy elevado para aplicaciones reales (donde hay un gran número de variables y se desea buena precisión).

Alander [3] determinó, basándose en evidencia empírica que, para problemas de complejidad moderada, el tamaño óptimo de la población para un GA con codificación binaria debía estar comprendido entre L y $2L$.

Para el caso de codificación real, en [20] se propone un número de individuos $5^L \leq N \leq 20^L$, siendo L la dimensión del espacio de búsqueda que produce buenos resultados, tanto en funciones de test, como en aplicaciones reales de identificación y control de procesos [71].

Otro aspecto importante es la inicialización de la población, la cual suele realizarse a partir de la generación de individuos aleatorios con probabilidad uniforme, siendo posible utilizar estrategias alternativas. Si bien estas estrategias pueden acelerar el proceso de convergencia del EA, podría ser que la convergencia fuese prematura y la solución obtenida fuera un mínimo local.

El objetivo de la creación de la población inicial es que ésta cubra adecuadamente el espacio de búsqueda, de ahí que una alternativa podría ser distribuir la población a lo largo del espacio de búsqueda utilizando un *grid* regular. La diferencia, con respecto a la inicialización aleatoria, es que ésta última genera nuevas posiciones dentro del espacio de búsqueda al repetir la ejecución del algoritmo, incluso podría generar soluciones no esperadas por el programador (especialmente si la estrategia utilizada es GP).

Otra posibilidad, es inicializar la población utilizando el conocimiento de la función de *fitness* que, haría que la población se concentrase en alguna zona especifica del espacio de búsqueda[8]. La figura 2.3 muestra gráficamente los ejemplos de inicialización mencionados.

[7] Fenómeno por el cual un EA converge en pocas generaciones hacía un punto del espacio de búsqueda.

[8] Es preferible esta solución que la reducción del espacio de búsqueda por si la solución no se encontrarse en la zona cubierta por la población inicial.

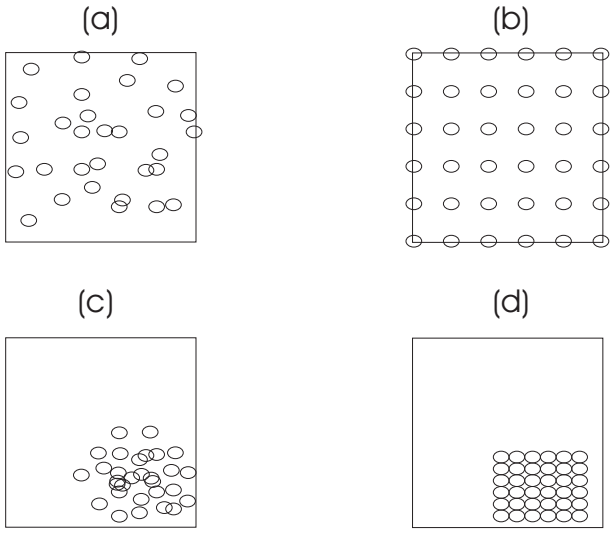

Figura 2.3: Ejemplo de inicialización de $P(0)$ para un espacio bidimensional. (a) inicialización aleatoria, (b) inicialización en *grid*, (c) y (d) inicialización basada en conocimiento, aleatoria y en *grid* respectivamente.

2.5. Operadores de selección y determinación

El operador de selección se encarga de escoger aquellos individuos que van a evolucionar y el de determinación aquellos que sobrevivirán, y por lo tanto, pasarán a formar parte de la población en la siguiente generación.

Los operadores de selección y determinación juegan un papel fundamental dentro del proceso evolutivo de los EAs y normalmente uno condiciona el otro, es decir, un operador de selección acarrea su operador de determinación y generalmente dependerán del tipo de algoritmo a utilizar (EP, ES, GA).

Normalmente, en el caso de los EPs y ESs la selección es aleatoria, mientras que es la determinación la que juega el papel más importante. En los GAs, por contra, es la selección la que toma protagonismo a costa de la determinación que desaparece. De ahí que se hable, de forma genérica, de un proceso de selección, tanto para describir el proceso de selección de la figura 2.1 cuando se utiliza un algoritmo tipo GA, como para describir el proceso de determinación, cuando se utiliza un EP o ES.

Las diferentes técnicas de selección las podemos clasificar como:

- **Estáticas o Dinámicas**: Las estáticas mantienen la probabilidad de selección constante a lo largo de las generaciones, mientras que las dinámicas no.

- **Preservativas o Extintivas**: Las preservativas requieren una probabilidad de selección distinta de cero para todos los individuos y las extintivas pueden asignar un probabilidad de selección de cero a algún individuo.

- **Solapables**: En un EA solapable los padres compiten con sus hijos [9].
- **Generaciones**: En los EAs generacionales la población entera es reemplazada y en los no Generacionales, sólo una parte de la población es reemplazada.
- **Elitistas**: Un EA elitista es aquel que garantiza que el mejor individuo de la población no desaparecerá en la siguiente generación.

Dos aspectos son importantes dentro del proceso evolutivo relacionados con el proceso de selección: la diversidad de la población y la presión de selección. El primero de los aspectos, la *diversidad*, hace referencia a la capacidad que tiene el algoritmo de explorar el espacio de búsqueda y no concentrarse rápidamente en zonas determinadas (convergencia prematura). La *diversidad* es importante especialmente en optimización de funciones multimodales o multiobjetivo (donde existen varios mínimos locales y/o globales). El concepto de *presión de selección*, por contra, hace referencia a la capacidad de convergencia de algoritmo. Ambos conceptos, *diversidad* y *presión de selección* están en contraposición, ya que una *presión de selección* fuerte fomentará la convergencia prematura y una débil provocará una búsqueda inefectiva. Por lo tanto, habrá que encontrar un equilibrio entre ambos aspectos a través de la utilización de los diferentes métodos de selección.

A continuación, se procede a describir los métodos de selección más utilizados en los EAs. La selección por torneo, la proporcional (con su variante utilizando *ranking*) y la de estado uniforme (o *steady-state*).

2.5.1. Selección proporcional

El operador de selección proporcional escoge con mayor probabilidad aquellos individuos que presentan un valor de la función de $fitness$ mejor, incluyendo además la posibilidad de que individuos que no son tan buenos tengan cierta probabilidad de ser seleccionados. Esto último, pretende mantener una cierta diversidad en cuanto a la información que contienen los cromosomas. No es bueno que desaparezcan rápidamente todos los individuos malos puesto que pueden contener parte de la información genética necesaria (parte de un cromosoma) para obtener el óptimo.

El objetivo que se persigue con la selección proporcional es que la cantidad de cada uno de los individuos de la nueva población venga dado por la siguiente proporción[10]:

$$Prob(p^i) = \frac{J(p^i)}{\sum_{j=1}^{N} J(p^j)}, \tag{2.5}$$

$$\sum_{j=i}^{N} Prob(p^i) = 1.$$

[9] Se denomina *brecha generacional* a la cantidad de solape existente entre padres e hijos.

[10] Esta expresión es válida para maximizar $J(\cdot)$ con $J(\cdot) > 0$. En caso de tener valores de $J(\cdot) < 0$ habría que sumar un término constante para convertirlos en positivos. Para el caso de minimización es posible utilizar la misma expresión invirtiendo los valores de $J(\cdot)$.

Donde p^i representa un individuo presente en la población de la generación anterior $P(t-1)$ y $J(p^i)$ representa su valor de la función de *fitness*. Este método de selección también se conoce como método de la ruleta simple [93].

La parte fija de la ruleta esta dividida en porciones que corresponden a los distintos tipos de cromosomas que existen en la población inicial. Por ejemplo, la población antes de la selección está compuesta por 12 individuos de los cuales sólo existen 8 tipos diferentes (cromosomas diferentes). Las porciones de la parte fija de la ruleta no son iguales, dependen del valor de la función de *fitness* del individuo al que representan según la expresión 2.5. Para obtener todos los individuos de la nueva población se hace girar la ruleta tantas veces como individuos se quieran seleccionar.

Por ejemplo, para obtener una nueva población de 12 individuos se hace girar 12 veces la ruleta, el puntero indica el tipo de individuo que se selecciona en cada lanzamiento.

Este mecanismo no asegura que la siguiente generación que se obtiene mediante la selección cumpla las proporciones de la expresión 2.5, puesto que depende de los lanzamientos que se realicen.

La solución pasa por utilizar una ruleta que no tenga un único puntero, sino tantos como individuos se quieren seleccionar para la siguiente generación. Los punteros deben estar uniformemente distribuidos a lo largo de la ruleta (ver figura 2.4). Haciendo girar una única vez la ruleta se obtienen todos los individuos de la población. Con este mecanismo, conocido como *Stochastic Universal Sampling* [14] sí que se asegura que la probabilidad de selección de un individuo es el que corresponde a la expresión 2.5.

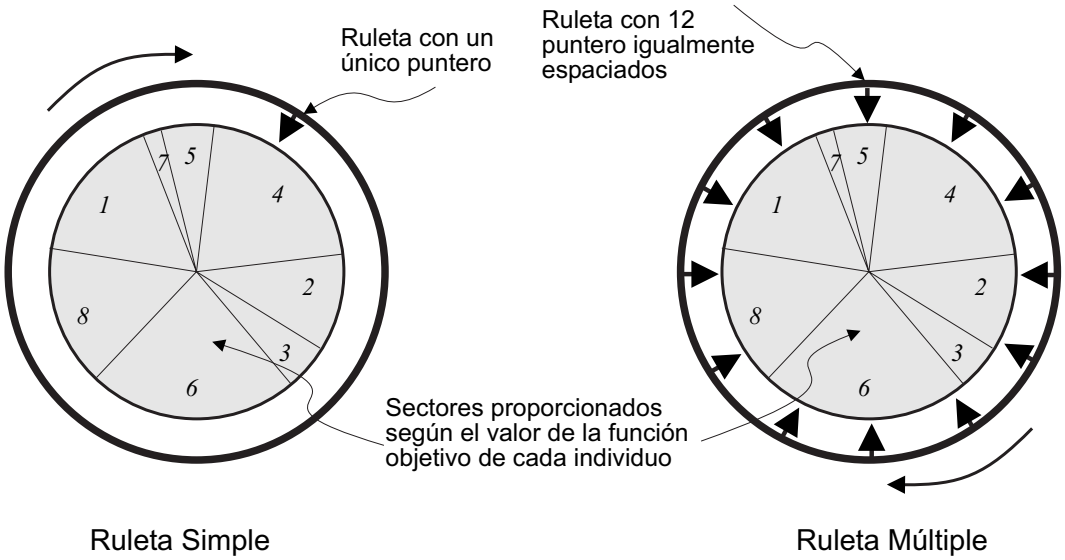

Figura 2.4: Representación gráfica de los mecanismos de selección de ruleta simple y múltiple. Las porciones de las partes fijas de ambas ruletas satisfacen la expresión $\frac{J(p^i)}{\sum_{j=1}^{N_{ind}} J(p^j)}$.

Atendiendo a la clasificación previa, se trata pues de un método dinámico, preservista, no solapable, generacional y no elitista. Uno de los principales problemas de este método es

que ejerce una presión de selección demasiado fuerte que puede propiciar la convergencia prematura, incrementando notablemente el riesgo de que el algoritmo se bloquee en un óptimo local. Este fenómeno ocurrirá cuando existan individuos con un valor de la función de coste mucho mejor que los demás, el operador de selección escogerá muchos individuos de este tipo para la siguiente generación disminuyendo las posibilidades para otros tipos de individuos no tan buenos.

2.5.1.1. Selección por ranking

Para conseguir un adecuado funcionamiento del algoritmo, se deben evitar convergencias prematuras del mismo, es decir, que rápidamente toda la población este compuesta por un único tipo de individuo. Una solución a este problema es modificar el valor de la función de *fitness* haciendo que los valores muy buenos y muy malos no lo sean tanto.

Este efecto se puede conseguir mediante una operación previa a la selección, denominada **operación de Ranking** [13]. Ésta consiste por ejemplo, en ordenar los individuos según el valor de la función de *fitness* y asignar como nuevo valor de la función de *fitness* el orden en esa clasificación. Esta operación se conoce como *ranking* lineal y, convierte al método en estático.

La tabla siguiente muestra un ejemplo de *ranking* lineal para un problema de minimización.

Individuo	Valor función de coste	Valor función de coste con *Ranking*
p^1	3.73	4
p^2	0.012	6
p^3	50.3	1
p^4	2.145	5
p^5	15.12	3
p^6	25.72	2

Evidentemente, ésta no es la única forma de reajustar los valores de la función de *fitness*, se podrían realizar otros tipos de asignaciones de valores que no sean lineales.

El problema de esta técnica, reside en el coste computacional requerido para ordenar la población, que puede volverse significativo en poblaciones grandes.

2.5.2. Selección por torneo

La selección por torneo [161] ejerce una presión de selección similar al método probabilístico con *ranking*, aunque requiere un coste computacional menor y puede paralelizarse fácilmente.

La técnica consiste en escoger, al azar, un número n de individuos de la población $P(t-1)$ y someterlos a un torneo seleccionando el mejor individuo de este grupo, si se usa

su variante determinística[11]. Se repite el proceso anterior, hasta que la población $P'(t)$ contiene N individuos.

Este método es fácil de implementar, produce buenos resultados y necesita pocos parámetros, lo que ha provocado que sea el método más usado hoy en día.

Modificando el valor n, es posible cambiar la presión de selección:

- $n = 1$, la selección es totalmente aleatoria y la presión de selección es muy baja.
- $2 \leq n \leq 5$, la presión de selección se considera blanda.
- $n \geq 10$, la presión se considera dura.

Este método es extintivo, no solapable, generacional y no elitista, aunque en media, el método asegura que el mejor individuo es mantenido en la población con un grado de repetición n (n copias del mejor individuo).

2.5.3. Selección de estado uniforme

La técnica de selección de estado uniforme (*steady state selection*) [162] actualiza sólo una pequeña cantidad de la población en cada generación, es decir, es un método no generacional. Suele utilizarse en ESs, aunque también en los GAs no generacionales.

El procedimiento crea n individuos (que conformarán la población $P'(t)$) a partir de individuos escogidos aleatoriamente de la población $P(t-1)$ y posteriormente alterados. A continuación, de entre los $n+N$ individuos se escogen los mejores para componer $P(t)$. Un valor típico para n es $0.15 * N$.

El método es extintivo, solapable, no generacional y elitista, pues garantiza que los mejores individuos sobreviven.

En general, esta técnica de selección resulta especialmente útil cuando la población resuelve un problema de forma colectiva (caso multiobjetivo o multimodal) dado su carácter no generacional y se usa cuando es importante recordar lo que se va aprendiendo durante las generaciones anteriores.

2.6. Operadores genéticos

Una vez definida la estructura de los cromosomas y una operación de decodificación, ya se puede seleccionar el conjunto de operadores genéticos que se aplicarán sobre los individuos de la población. Como ya se comentó, estos operadores dependerán del tipo de

[11] En el caso que $n = 2$ se puede usar la versión probabilística donde el torneo lo gana el mejor individuo con probabilidad α o el peor con probabilidad $(1-\alpha)$, $0.5 < \alpha < 1$.

codificación. A continuación, se presentan diferentes versiones de los operadores de cruce y mutación[12] para los dos tipos de codificaciones binaria y real presentadas anteriormente.

2.6.1. Operadores de cruce

El objetivo de esta operación es el de mezclar la información de los cromosomas con el fin de obtener individuos diferentes a los que ya existen. Se trata de explorar el espacio de búsqueda combinando información existente. La operación de cruce no suele realizarse sobre todos los individuos de la población, cada individuo tiene una probabilidad de ser cruzado, probabilidad de cruce p_c. Si se cruzan todos los individuos, cada generación podría resultar totalmente diferente de la anterior[13] y el algoritmo podría tener dificultades para converger. Para la operación de cruce existen varias posibilidades, en parte dependientes del tipo de codificación empleada.

2.6.1.1. Codificación binaria

Para el caso de la codificación binaria es posible utilizar, entre otros, los siguientes operadores de cruce:

- **Cruce simple**. En el cruce simple o cruce de un punto [83] se selecciona aleatoriamente una posición por donde se van a cortar los cromosomas padres para combinarse (figura 2.5).

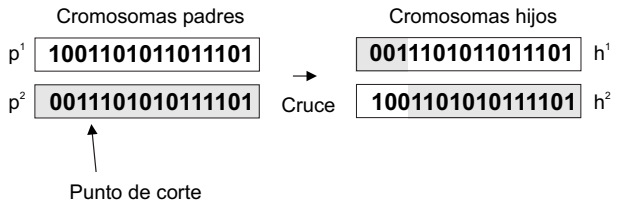

Figura 2.5: Cruce simple en la codificación binaria.

- **Cruce multipunto**. En el cruce multipunto [93] los individuos hijos se obtienen de cortar los cromosomas de los padres por varios puntos en lugar de por uno sólo (figura 2.6). Esto permite una mayor variedad de cromosomas hijos y más posibilidades de explorar el espacio de búsqueda, sin embargo, aumenta la probabilidad de romper excesivamente buenos cromosomas.

- **Cruce uniforme**. Uno de los problemas que pueden presentar los algoritmos genéticos es la convergencia prematura del algoritmo, debido básicamente a una pérdida

[12]El papel que juega cada uno de estos operadores, así como su comparación, es un tema aún por resolver dentro de la comunidad de computación evolutiva.

[13]Esto no ocurre si el método de selección es de estado uniforme, ya que sólo una parte de la población es modificada, pudiéndose utilizar $pc = 1$.

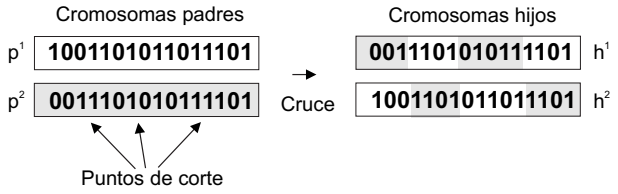

Figura 2.6: Cruce multipunto en la codificación binaria.

de diversidad de la población (los individuos presentes en la población tienen cromosomas muy similares, por tanto es difícil que varíen mucho los cromosomas de la siguiente generación). Con el cruce uniforme se pretende ampliar las posibilidades de combinación de los cromosomas padre para generar cromosomas hijos.

El cruce uniforme [151, 94] puede considerarse como una extensión del *cruce multipunto*. En el cruce uniforme, los cromosomas hijos se obtienen de la siguiente forma: cada uno de los bits del cromosoma del primer hijo puede obtenerse de un cromosoma padre u otro dependiendo de una determinada probabilidad (α).

En la figura 2.7, las cadenas *origen* o^1 y o^2 indican de qué padre se generan cada uno de los bits de los hijos, si el valor es 1 significa que el bit se toma del padre 1 (p^1), si el valor es 2 el bit se toma del padre 2 (p^2). Cada una de las cadenas corresponde al origen de uno de los hijos y se han generado aleatoriamente con una probabilidad α, la primera cadena o^1 corresponde a h^1 y la segunda o^2 a h^2 siendo la inversa de o^1. Dependiendo de la implementación que se realice o^2 no tiene por qué ser complementario a o^1, en este caso o^2 se genera con una probabilidad $1-\alpha$. Esta implementación es un poco más costosa computacionalmente, ya que se deben generar más números aleatorios.

Para $\alpha = 0.5$ los cromosomas hijos tienen (en promedio) tanta información genética de un padre como del otro. Si se aumenta (o disminuye) el valor de α los cromosomas hijos contienen más información genética de uno de los padres (se va a parecer más a ese padre).

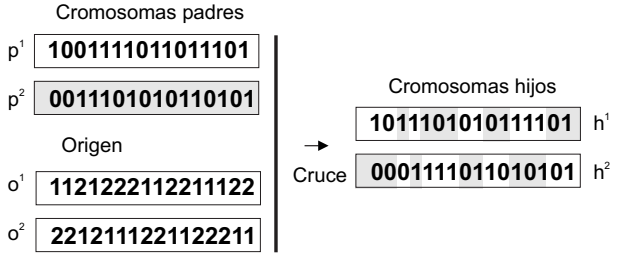

Figura 2.7: Cruce uniforme en la codificación binaria.

2.6.1.2. Codificación real

Para realizar las operaciones de cruce con codificación real se pueden utilizar las operaciones que se han definido para la codificación binaria, siempre que el espacio de búsqueda sea de dimensión mayor o igual a dos (para una dimensión sólo se pueden aplicar los operadores de cruce por recombinación que se describen más adelante):

- Cruce simple.
- Cruce multipunto.
- Cruce uniforme.

La forma de realizar estas operaciones es utilizar cada una de las variables del cromosoma de la codificación real como un bit de la codificación binaria, por tanto los cruces se producirán en las posiciones que separan las variables (figura 2.8).

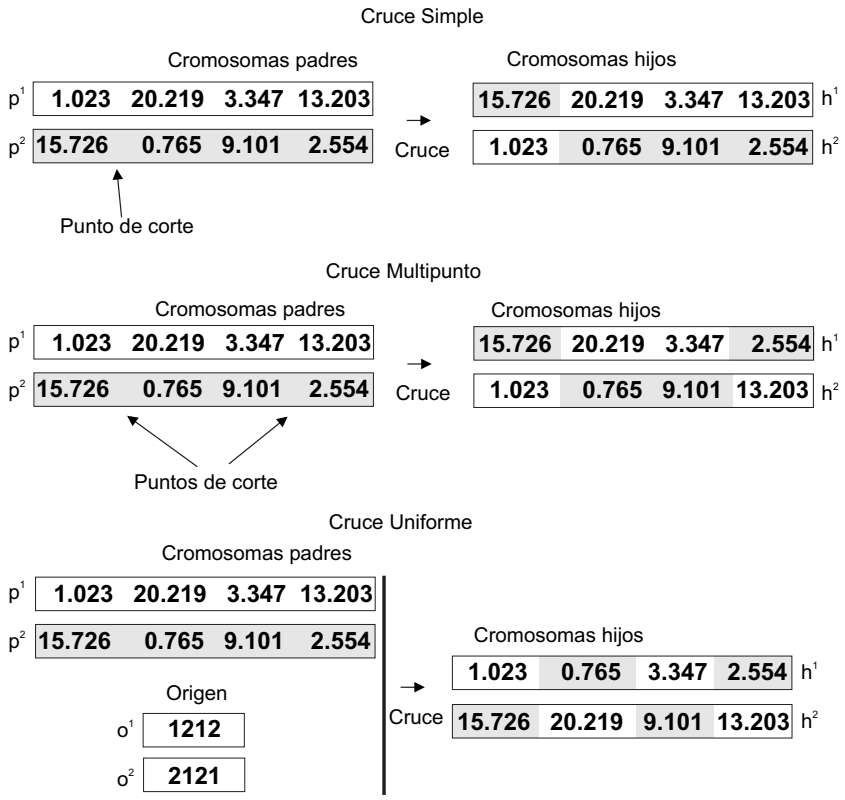

Figura 2.8: Operaciones de cruce en la codificación real. Cruce simple, multipunto y uniforme.

Con la codificación real aparece otro conjunto de posibilidades para realizar la operación de cruce que permiten, además, una mayor diversidad en la población resultante si se compara con los operadores de cruce anteriores, además, estos operadores de cruce sí que se pueden utilizar para un espacio unidimensional.

- **Recombinación lineal**. Los cromosomas hijos se obtienen realizando una combinación lineal de los cromosomas padre, se aplica sobre cada variable del cromosoma las siguientes operaciones:

$$h^1 = \alpha \cdot p^1 + (1-\alpha) \cdot p^2,$$
$$h^2 = (1-\alpha) \cdot p^1 + \alpha \cdot p^2.$$

  El valor del parámetro α es el mismo para todas las variables del cromosoma y se puede elegir en el intervalo $[-d, 1+d]$. Para la recombinación lineal se utiliza $d = 0$, para la recombinación lineal extendida un valor usual es $d = 0.25$.

  Si se asigna un valor de $d > 0$ se expande la zona donde pueden aparecer los hijos, se incrementa la propiedad de exploración del espacio de búsqueda, aunque un valor de d demasiado elevado dificulta la convergencia del algoritmo (la población se puede dispersar demasiado).

- **Recombinación intermedia**. Se realizan las mismas operaciones que en la recombinación lineal, la única diferencia es que se utiliza un valor diferente $\alpha_i,\ i \in [1, \ldots, L]$ para cada una de las variables.

$$\theta_i^{h1} = \alpha_i \cdot \theta_i^{p1} + (1-\alpha_i) \cdot \theta_i^{p2},$$
$$\theta_i^{h2} = (1-\alpha_i) \cdot \theta_i^{p1} + \alpha_i \cdot \theta_i^{p2}.$$

El efecto de este tipo de operadores de cruce se puede ver gráficamente (figura 2.9) para el caso de un espacio de búsqueda bidimensional, se muestran donde pueden aparecer los individuos resultantes del cruce que sean diferentes a los padres. En la figura 2.9 los cromosomas padres son:

$$p^1 = [\theta_1^{p_1}, \theta_2^{p_1}],$$
$$p^2 = [\theta_1^{p_2}, \theta_2^{p_2}].$$

Se puede ver cualitativamente la capacidad de exploración del espacio de búsqueda de cada uno de los operadores de cruce. Para un espacio bidimensional los cruces por punto simple, multipunto y uniforme son equivalentes, en la figura sólo se muestra uno de ellos (punto simple). Otro operador de cruce muy extendido es el de **cruce por simulación binaria (SBX)** [42] que trata de emular el cruce de punto simple. La operación de cruce se realiza de la siguiente manera:

$$\theta_i^{h^1} = 0.5 \cdot \left((1+\beta_i) \cdot \theta_i^{p^1} + (1-\beta_i) \cdot \theta_i^{p^2} \right)$$
$$\theta_i^{h^2} = 0.5 \cdot \left((1-\beta_i) \cdot \theta_i^{p^1} + (1+\beta_i) \cdot \theta_i^{p^2} \right),$$

donde β_i se determina de la siguiente manera:

$$\beta_i = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}} & \text{si } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}} & \text{otro caso} \end{cases},$$

siendo u_i un número aleatorio de distribución uniforme entre $[0 \ldots 1[$ y η_c un parámetro del operador. La figura 2.10 muestra el resultado de cruzar $p^1 = [0\ 0]$ y $p^2 = [1\ 1]$ con diferentes valores de η_c.

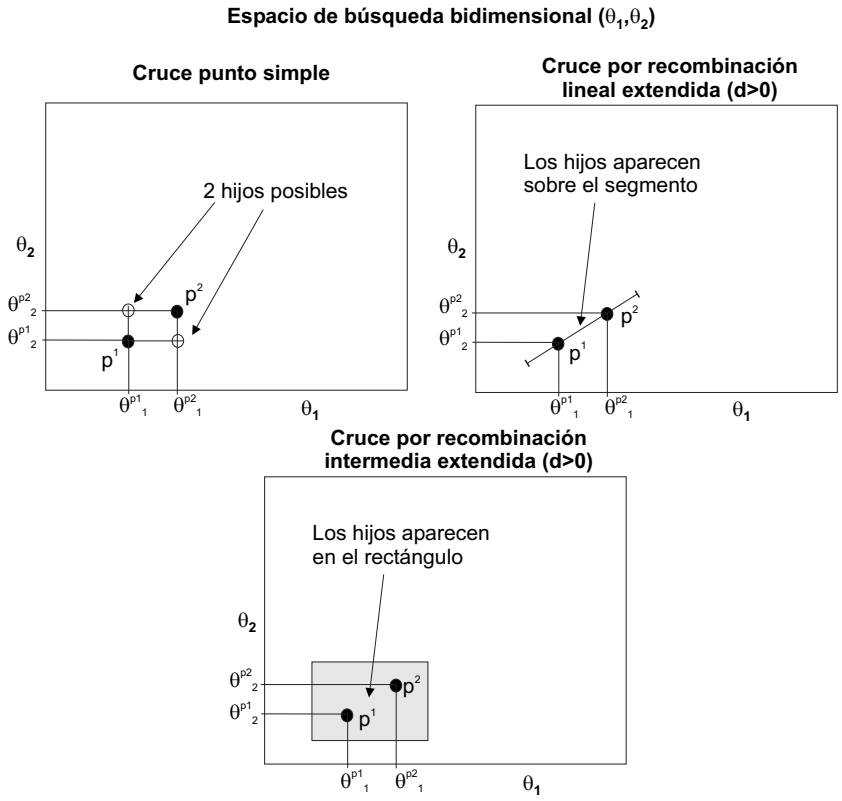

Figura 2.9: Efecto de las operaciones de cruce en la codificación real. Cruce simple y recombinaciones lineal e intermedia extendidas ($d > 0$).

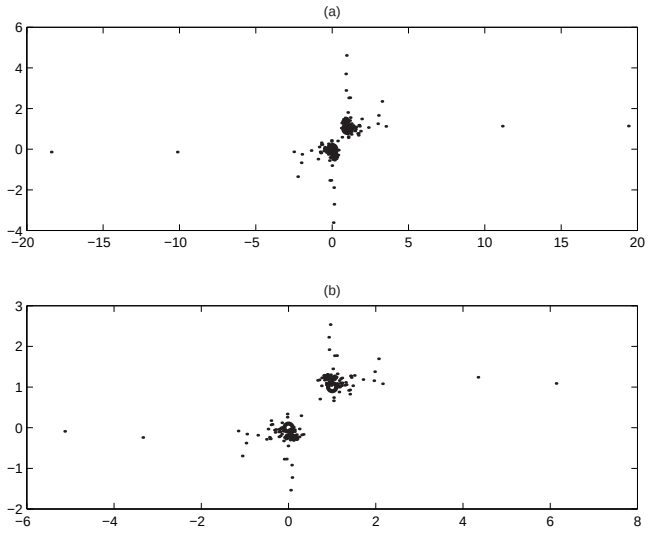

Figura 2.10: Resultado de cruzar los padres $p^1 = [0\ 0]$ y $p^2 = [1\ 1]$ utilizando cruce por simulación binaria con η_c (a) 1 y (b) 2, recomendados por los autores.

2.6.2. Operadores de mutación

Este operador permite incrementar la capacidad de exploración del espacio de búsqueda puesto que posibilita la aparición de información genética que no existe en la población. En este sentido se complementa al operador de cruce, el cual permite que aparezcan cromosomas nuevos pero combinando información genética de sus padres. El operador de mutación puede, aunque no siempre, hacer aparecer nueva información. La probabilidad de mutación p_m debe ser baja puesto que de lo contrario se dificulta la convergencia del algoritmo.

2.6.2.1. Codificación binaria

La variación aleatoria de los bits del cromosoma de los individuos de la población con una probabilidad p_m (figura 2.11) es el tipo de mutación más utilizado. Si varían aleatoriamente muchos bits de los cromosomas es muy difícil que los individuos se parezcan cada vez más, impidiendo la convergencia del algoritmo. En la práctica, se recomienda que p_m se encuentre entre 0.001 y 0.01 [20, 93]. Algunos autores [6, 93] sugieren que $p_m = 1/L$.

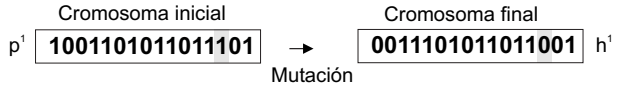

Figura 2.11: Mutación en la codificación binaria.

2.6.2.2. Codificación real

Igual que para la codificación binaria, el operador de mutación se encarga de incrementar la capacidad de exploración del espacio de búsqueda, en concreto trata de hacer que aparezca en la población información genética que no existía. Cuando se utilizan los operadores de cruce binarios esta operación de mutación es imprescindible para que aparezca nueva información genética, pero si los operadores que se utilizan son los de recombinación puede que no sea necesaria, aunque sí recomendable, ya que los operadores de recombinación (lineal o intermedia) son capaces de generar individuos con información genética diferente a la de los padres (especialmente en su versión extendida $d > 0$). Esta característica puede ser interesante desde el punto de vista de coste computacional y sencillez del algoritmo.

La forma más común de realizar la mutación de un parámetro es la **mutación aleatoria** que consiste en añadir un valor aleatorio, con una distribución uniforme, a dicho parámetro manteniendo el parámetro resultante en el espacio de búsqueda.

$$\theta_i^h = \theta_i^p + U(-\alpha, \alpha).$$

El comportamiento del operador de mutación dependerá en gran mediada del parámetro α que afectará a la convergencia del algoritmo. Valores de α elevados dificultarán el ajuste fino de las soluciones, mientras que valores pequeños reducirán la capacidad de generar diversidad, facilitando la captura de un óptimo local.

Por ello, en ocasiones, es conveniente que el operador de mutación provoque variaciones pequeñas (el parámetro mutado se mantiene cerca del parámetro original) para tratar de aumentar la precisión de la solución encontrada y en otras ocasiones interesa justamente lo contrario para explorar nuevas zonas.

El operador **mutación orientada** [129] provoca con mayor probabilidad mutaciones de valores bajos, aunque también ocurren mutaciones con valores altos:

$$\theta_i^h = \theta_i^p \pm 0.5 \cdot (\theta_{i\,max} - \theta_{i\,min}) \cdot \delta.$$

Donde:

$$\delta = \sum_{i=1}^{m} \alpha_i 2^{-i}$$

$$\alpha_i = \begin{cases} 1 & \text{con una probabilidad } \frac{1}{m} \\ 0 & \text{en el resto de los casos} \end{cases}$$

$$m = 20.$$

Se puede conseguir un efecto intermedio si se utiliza un operador de mutación aleatoria pero con una distribución distinta de la uniforme, por ejemplo, con una distribución normal (ver figura 2.12).

$$\theta_i^h = \theta_i^p + N(0, \alpha).$$

Otro operador, con resultados similares al de mutación con distribución normal (ver figura 2.12) es el de **mutación polinomial**, cuyo procedimiento es el siguiente:

$$\theta_i^h = \theta_i^p \pm (\theta_{i\,max} - \theta_{i\,min}) \cdot \delta_i.$$

donde δ_i se determina de la siguiente manera:

$$\delta_i = \begin{cases} (2u_i)^{\frac{1}{\eta_m+1}} - 1 & \text{si } u_i < 0.5 \\ 1 - (2(1-u_i))^{\frac{1}{\eta_m+1}} & \text{otro caso} \end{cases},$$

siendo u_i un número aleatorio de distribución uniforme entre 0 y 1 y η_m un parámetro del operador que ejerce una función similar, pero en sentido inverso, a la varianza α del operador con distribución normal.

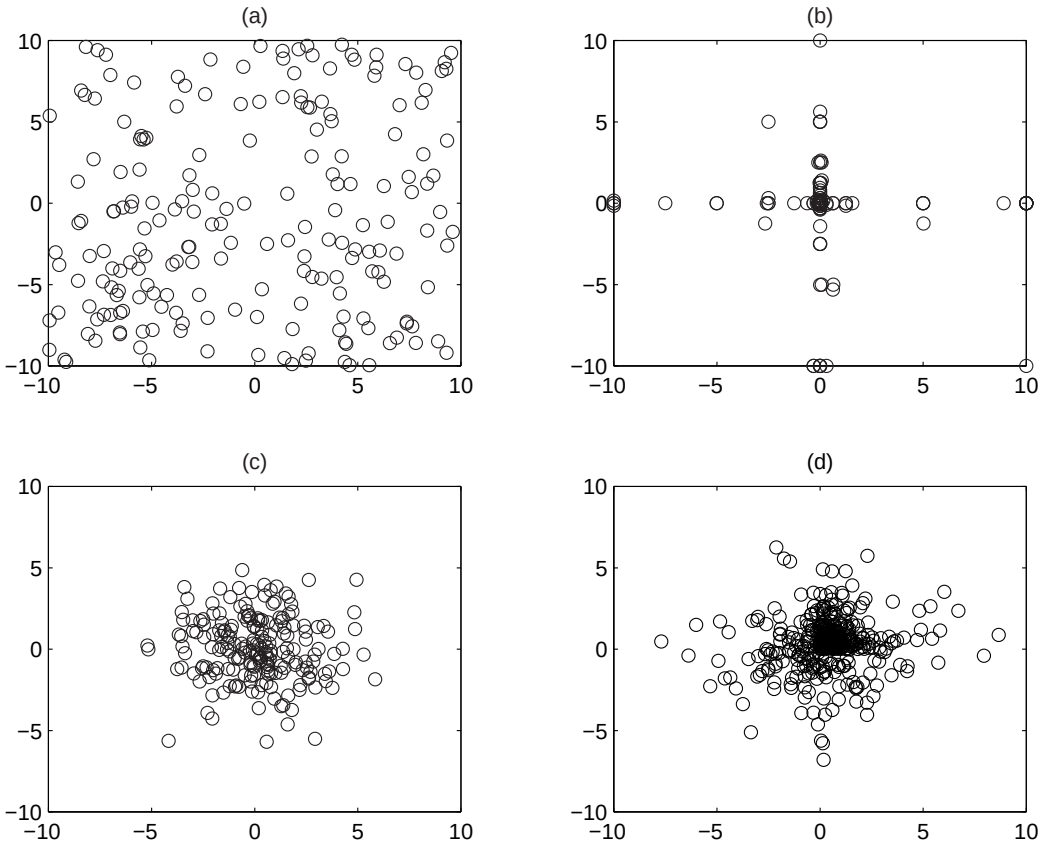

Figura 2.12: Mutaciones del punto (0,0) en el espacio $[-10, 10] \times [-10, 10]$ con distintos operadores de mutación. (a) Mutación aleatoria con distribución uniforme, (b) Mutación orientada, (c) Mutación aleatoria con distribución normal N(0,2) y (d) Mutación polinomial $\eta_m = 10$.

2.7. Aspectos de las funciones a optimizar

La construcción de la función de *fitness* juega un papel muy importante dentro del problema de optimización determinando, en gran medida, la dificultad de dicho proceso. Desde un punto de vista teórico, la función de *fitness* suele venir dada de forma explícita, por un conjunto de ecuaciones matemáticas. Por otra parte, en problemas de optimización reales[14] la función de *fitness* no suele estar definida y es parte de la tarea correspondiente al diseñador o usuario del EA y al experto en el área del problema a resolver.

El aspecto más importante que debe cumplir la función de *fitness* está relacionado con la asignación del mejor valor[15] a la solución deseada y, valores peores, a soluciones de peor calidad, con el fin de que se reduzcan sus posibilidades de supervivencia en generaciones posteriores.

Otros aspectos o características importantes a tener en cuenta en las funciones de *fitness* son:

- La forma.
- Los mínimos locales.
- Coste computacional.
- Múltiples objetivos.

La forma. Aspectos como zonas planas, irregularidades o valles en la función de *fitness* pueden dificultar la solución al problema de optimización (ver figura 2.13). En ocasiones estos aspectos pueden ser evitados simplemente transformado adecuadamente la función de *fitness*.

Las zonas planas o llanuras provocan que el proceso de selección no funcione, ya que en esa zona todos los individuos presentan el mismo valor para la función de *fitness* y el proceso de optimización se reduce a una búsqueda aleatoria, dado que no se dispone de información para direccionar adecuadamente la búsqueda.

Algo parecido ocurre cuando la función de *fitness* presenta muchas irregularidades. En este caso, un punto del espacio de búsqueda y sus vecinos podrían presentar valores arbitrarios de la función de *fitness*. Aunque este tipo de problemas no suelen ser habituales en problemas de optimización numérica, ya que las variables suelen ser continuas[16], pueden aparecer en ocasiones debido a problemas de precisión numérica. Por ejemplo, si para determinar el valor de la función de *fitness* es necesario aplicar integración numérica y se utiliza un paso de integración variable, las aproximaciones que incorpora la integración podrían producir irregularidades [155].

[14] Por ejemplo, identificación y control de procesos.

[15] El mínimo si se está minimizando y el máximo si se está maximizando.

[16] No ocurre lo mismo en GP donde la codificación de las soluciones puede provocar que pequeñas modificaciones en el individuo generen valores de *fitness* muy dispares.

Los valles o surcos en la función de *fitness* suelen ser difíciles de superar, ya que un individuo sólo mejorará si se mueve en la única dirección adecuada que existe, la que indica el fondo del valle. Esto significa que una vez un individuo ha alcanzado el valle, para alcanzar el fondo de éste, se han de modificar varias coordenadas del individuo al mismo tiempo y en la dirección adecuada. Este tipo de problemas suele aparecer cuando las variables del problema de optimización están correladas y la solución pasa por reescribir la función de *fitness*.

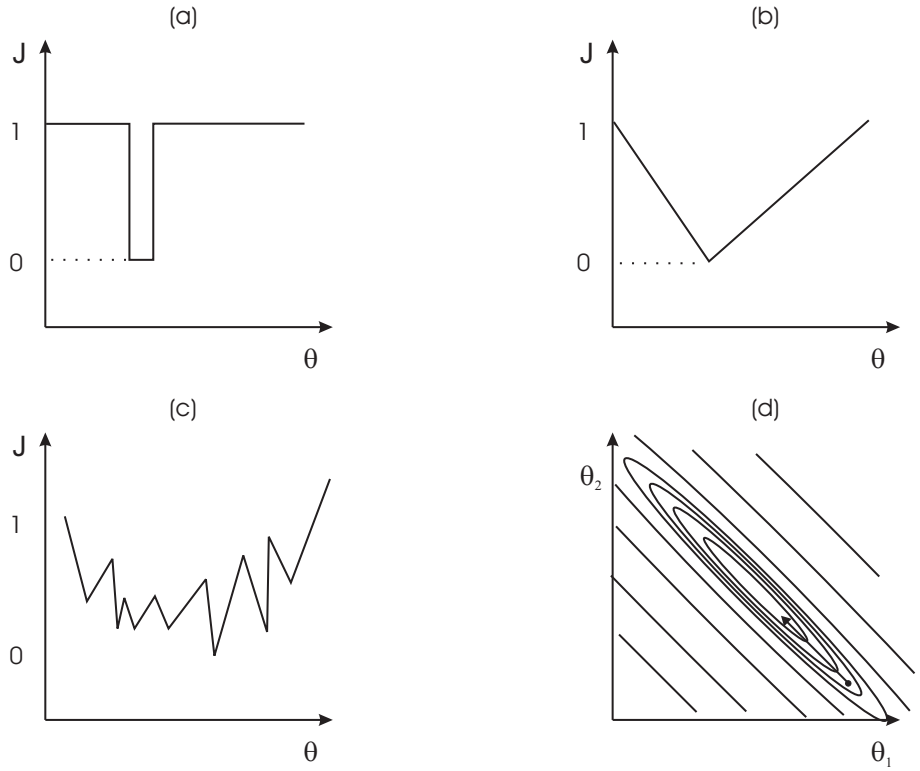

Figura 2.13: (a) Función de *fitness* con zonas planas. (b) Función modificada para eliminar las llanuras. (c) Función de *fitness* con irregularidades. (d) Curvas de nivel de una función con un valle. La flecha indica la única dirección en la que se puede mover el individuo si quiere mejorar su adaptación al medio (reducir el valor del la función de *fitness*).

Los mínimos locales suelen ser difíciles de eliminar de las funciones de *fitness* y son particularmente problemáticos, pues pueden provocar que el algoritmo de búsqueda quede enclavado en uno de ellos confundiéndolo con uno global. A estos problemas de optimización, donde hay más de un óptimo local, se les conoce como multimodales mientras que aquellos que sólo tienen un único mínimo local (y por lo tanto global) son conocidos como unimodales (ver figura 2.14).

Hay una serie de características de los EAs que les hacen estar especialmente indicados para afrontar problemas de optimización multimodales.

- Exploran el espacio de búsqueda desde varios puntos al mismo tiempo.

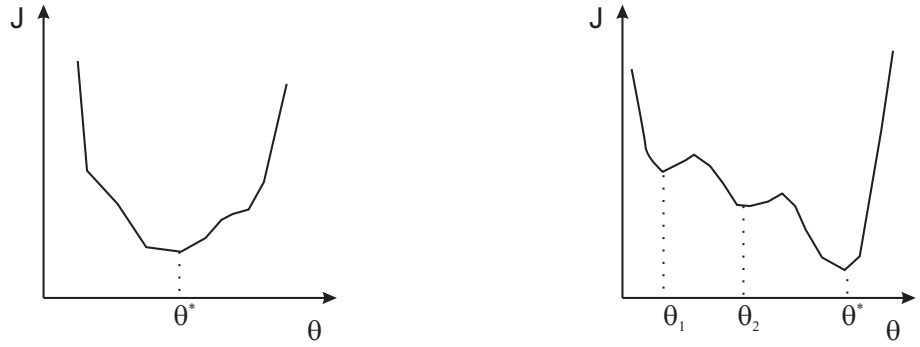

Figura 2.14: A la izquierda una función unimodal donde θ^* es el óptimo global. A la derecha una función multimodal con dos mínimos locales θ_1 y θ_2 y un mínimo global θ^*.

- Las soluciones que se van creando, no tiene porque estar necesariamente alrededor de otras soluciones existentes.

Dada la importancia y problemática asociada a los mínimos locales, muchos trabajos y técnicas específicas han sido creados con el objetivo de evitarlos. La sección 2.11 presenta un resumen de las técnicas más importantes.

El coste computacional es uno de los inconvenientes asociados a los EAs, ya que incrementa el tiempo que cuesta obtener la solución al problema. Principalmente, el coste computacional recae sobre el número de evaluaciones de la función de *fitness*[17] que es necesario hacer. Para paliar este problema se suelen aplicar tres enfoques diferentes:

- Simplificar la función de *fitness*, compactando operaciones incluso realizando aproximaciones de ésta (metamodelos).
- Almacenar las soluciones y el *fitness* que se van calculando con la esperanza de poder reutilizarlas posteriormente. El problema de este método es el elevado espacio de memoria que requiere.
- Utilizar EAs paralelos aprovechando la facilidad que presentan para ser paralelizados. De esta manera, se puede repartir el coste computacional entre varios ordenadores reduciendo el tiempo de cómputo global, en el mejor de los casos, dividiéndolo por el número de ordenadores utilizados. La sección 2.8 presenta las arquitectura de computación paralela más comunes utilizadas en los EAs.

Múltiples objetivos. En muchos problemas de la vida real se desea optimizar varios objetivos al mismo tiempo, por ejemplo, minimizar consumos maximizando prestaciones. Estos objetivos, generalmente contradictorios, pueden abordarse con la técnicas de optimización clásicas, incorporándolos de forma ponderada en un único índice o función de

[17]Por ejemplo, en identificación para determinar el valor de la función de *fitness* es necesario simular modelos complicados de procesos durante un tiempo considerable, lo que desemboca en un elevado coste computacional.

fitness. La elección de los coeficientes de ponderación suele ser bastante problemática, ya que condiciona la solución al problema.

Otra posibilidad es tratar de forma independiente los objetivos y determinar un conjunto de soluciones que los optimicen utilizando técnicas de optimización multiobjetivo. Este enfoque constituye una nueva línea de investigación que está tomando mucha fuerza durante estos últimos años [36, 35]. Como consecuencia de ello, han surgido nuevas generaciones de EAs especialmente concebidos para resolver problemas de optimización multiobjetivo, los MOEAs a los que se dedica el capítulo 3.

2.8. Arquitecturas paralelas para los EA

De forma genérica se podría hablar de cuatro arquitecturas paralelas diferentes [26, 31]:

- Maestro-Esclavo o paralelización global.
- De grano grueso o de isla.
- De grano fino o de difusión.
- Híbridas.

Resulta difícil comparar las diferentes arquitecturas, debido a su propia naturaleza, ya que podrían resultar injustas las comparaciones. En ocasiones la comparación se limita, simplemente, al análisis de la calidad de las soluciones encontradas.

2.8.1. Arquitectura maestro-esclavo

En esta arquitectura existe un procesador, el maestro, que ejecuta el EA y el resto, los esclavos, se encargan de evaluar la función de *fitness* actuando como si de servidores se tratasen (figura 2.15). El maestro se encarga de enviar los individuos de la población hacia los esclavos. Éstos, calculan el valor de la función de *fitness* del individuo recibido y devuelve su valor al maestro. Cuando el maestro ha recibido el valor de la función de *fitness* de todos los individuos de la población, efectúa una generación del EA y repite el proceso. A este tipo de procedimiento se le conoce como síncrono. Esta arquitectura es la más sencilla de implementar, ya que es equivalente a un EA convencional. Utiliza una población y los mismos operadores que un EA convencional, pero se puede obtener una reducción significativa del coste computacional[18] si los costes de evaluar la función de *fitness* superan a los de comunicación entre procesadores.

En ocasiones es posible paralelizar también los operadores genéticos, aunque no suele ser habitual, ya que las mejoras en reducción de tiempo no compensan los tiempos

[18]Si se dispone de m esclavos, cada uno ejecutado en un procesador diferente, será posible reducir el tiempo de computo idealmente m veces. En la práctica, esto no ocurrirá nunca y la eficacia dependerá de la buena implementación y la relación entre el tiempo de evaluación de J y los tiempos de comunicación.

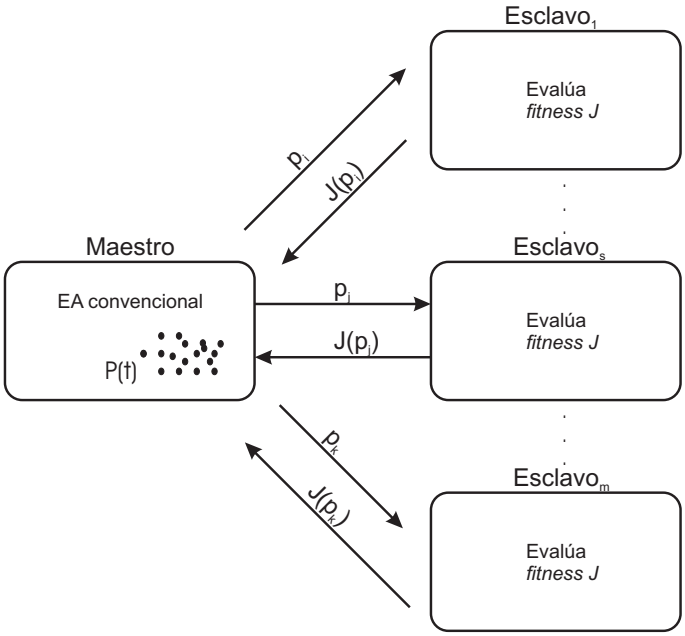

Figura 2.15: Arquitectura paralela EA maestro-esclavo.

de desarrollo. La Arquitectura Maestro-Esclavo puede implementarse también de forma asíncrona[19] pero, en este caso, el EA paralelo ya no resultaría totalmente equivalente al EA convencional.

2.8.2. Arquitectura de grano grueso

En esta arquitectura la población del EA es repartida entre m subpoblaciones que evolucionan de forma aislada (modelo de isla) intercambiando datos entre sí mediante un operador llamado *migración* (ver figura 2.16). El comportamiento del EA de grano grueso, también conocido como distribuido, es diferente al del EA convencional. Aunque la mayor parte de la programación del EA convencional es útil, es necesario implementar el nuevo operador de migración. Algunos de los parámetros adicionales de configuración del EA que aparecen son:

- Número de subpoblaciones o *demes*. Puede estar relacionado con el número de procesadores disponibles.
- Tamaño de las subpoblaciones $P_i(t)$, $i \in [1, ..., m]$.
- Topología de interconexión entre subpoblaciones. Juega un papel muy importante en el comportamiento del EA paralelo. Por ejemplo, una topología densa (con muchas

[19]El maestro no se detiene a esperar que la función de *fitness* de toda la población esté calculada. De esta manera, evita que procesadores lentos entorpezcan la marcha del EA.

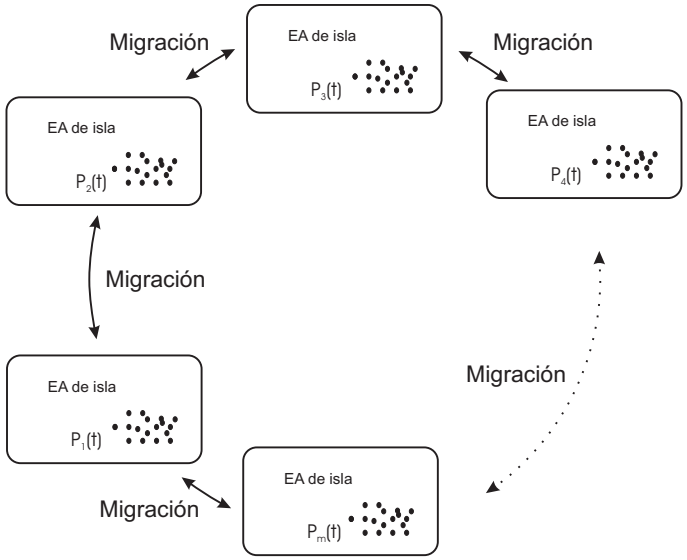

Figura 2.16: Arquitectura de grano grueso con topología en anillo.

interconexiones) promoverá una mejor mezcla de individuos entre *demes* aumentando al mismo tiempo los coste computaciones, mientras que una dispersa (con pocas interconexiones) los individuos se diseminarán más lentamente aislando los *demes* y favoreciendo la diversidad. Las topologías más comunes de interconexiones son en anillo, en árbol, en estrella o en red (ver figura 2.17). Un concepto relacionado con la topología es el de *vecindario*. Se refiere al área en el cual puede moverse un individuo que migre de un determinado *deme*.

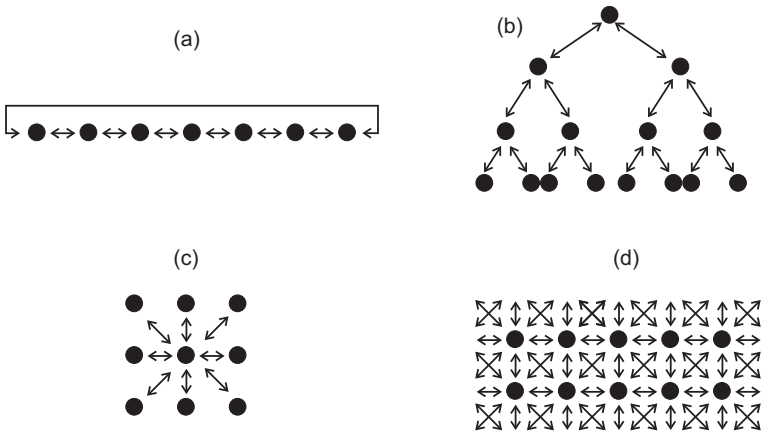

Figura 2.17: Topologías de conexión entre demes. (a) Conexión en anillo. (b) Conexión en árbol. (c) Conexión en estrella. (d) Conexión en red.

- Parámetros relacionados con la migración.
  - Intervalo de migración. Ratio entre iteraciones en las que se produce migración y el total de iteraciones.

- Tasa de migración. Ratio entre individuos que migran y el tamaño de las subpoblaciones.
- Radio de selección. Se refiere a la cantidad de *demes* desde los que se podrán seleccionar individuos desde un determinado *deme*. Normalmente, el radio de selección será cero y sólo se podrán seleccionar individuos de la propia subpoblación.
- Radio de migración. Está relacionado con la cantidad de *demes* a los que podrá migrar un individuo de un determinado *deme* o subpoblación. Evidentemente esto dependerá de la topología utilizada.

2.8.3. Arquitectura de grano fino

La idea es la misma que en la arquitectura de grano grueso, la diferencia es que ahora el número de subpoblaciones es mayor y el número de individuos de dichas subpoblaciones en bastante menor, idealmente uno. Cada subpoblación puede implementarse en un procesador diferente, dando pie a arquitecturas masivas en paralelo de procesadores o también pueden ejecutarse en un multiprocesador.

El problema de esta arquitectura recae sobre el coste de comunicaciones entre procesadores que puede llegar a degradar, de forma importante, las mejoras en prestaciones deseadas para una arquitectura paralela. La topología más habitual para este tipo de arquitectura es la de malla (ver figura 2.18).

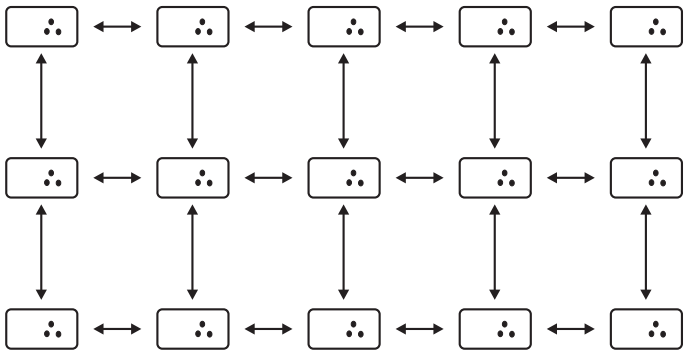

Figura 2.18: Tipología de conexión tipo malla utilizada en arquitecturas de grano fino.

2.8.4. Arquitectura híbrida

Este tipo de arquitectura es una mezcla de las arquitecturas anteriores, de ahí su nombre. La idea es diseñar una estructura que combine los beneficios de las anteriores sin llegar a complicarla demasiado. Generalmente, se suelen escoger dos tipos de estructuras combinándolas en dos niveles diferentes, alto y bajo (ver figura 2.19). Algunas combinaciones existentes son:

- Usar un EA de grano fino a bajo nivel y por encima otra de grano grueso.
- Usar un EA de grano grueso a alto nivel y por debajo un EA maestro-esclavo.
- Usar un EA de grano grueso tanto para el nivel alto como el bajo. La diferencia es que a nivel bajo la velocidad de migración es mayor y la topología es más densa.

2.9. Ajuste de los parámetros de los EA

Una de las dificultades más importantes cuando se trabaja con EAs, una vez ya se tiene definida la función de coste, es determinar los valores de los parámetros de configuración del mismo. Generalmente, cada parte del EA tiene un parámetro a ajustar: tamaño de la población N, probabilidad de cruce p_m, probabilidad de mutación p_c, etc. Estos parámetros normalmente interactúan entre sí de forma no lineal, de manera que no podrán optimizarse independientemente. Además, el ajuste óptimo de parámetros dependerá del problema a resolver, no existiendo un conjunto de parámetros que actúe adecuadamente para todo tipo de problemas [164]. De hecho los parámetros óptimos, seguramente, no deberían ser contantes durante el proceso de búsqueda.

Desde los orígenes de los EAs, los investigadores han desarrollando técnicas de ajuste de los parámetros, sin existir, hasta la fecha, una solución general. Una posible clasificación de estas técnicas sería la planteada en [155] donde se divide a éstas en:

- **No adaptativas**. El ajuste de los parámetros se hace sin considerar el estado o evolución del proceso de búsqueda.
- **Adaptativas**. Utilizan información del proceso de búsqueda para ajustar los parámetros de forma dinámica, mientras éste se produce.

El uso de técnicas adaptativas puede mejorar las prestaciones de las no adaptativas, sin embargo, no se evita totalmente el ajuste de parámetros. Lo que ocurre es que se pasa de ajustar los parámetros a ajustar los métodos de ajuste de parámetros. La ventaja es que esto último resulta, por lo general, más sencillo y menos sensible.

A continuación, se procede a mostrar las diferentes técnicas o estrategias utilizadas en los ajustes adaptativos y no adaptativos.

2.9.1. Métodos de ajuste no adaptativos

Dentro de este grupo de métodos o técnicas no adaptativas, se puede a su vez realizar una subdivisión separando entre:

- **Métodos de ajuste constante**. Determinan el valor adecuado de un parámetro para toda la evolución del algoritmo.

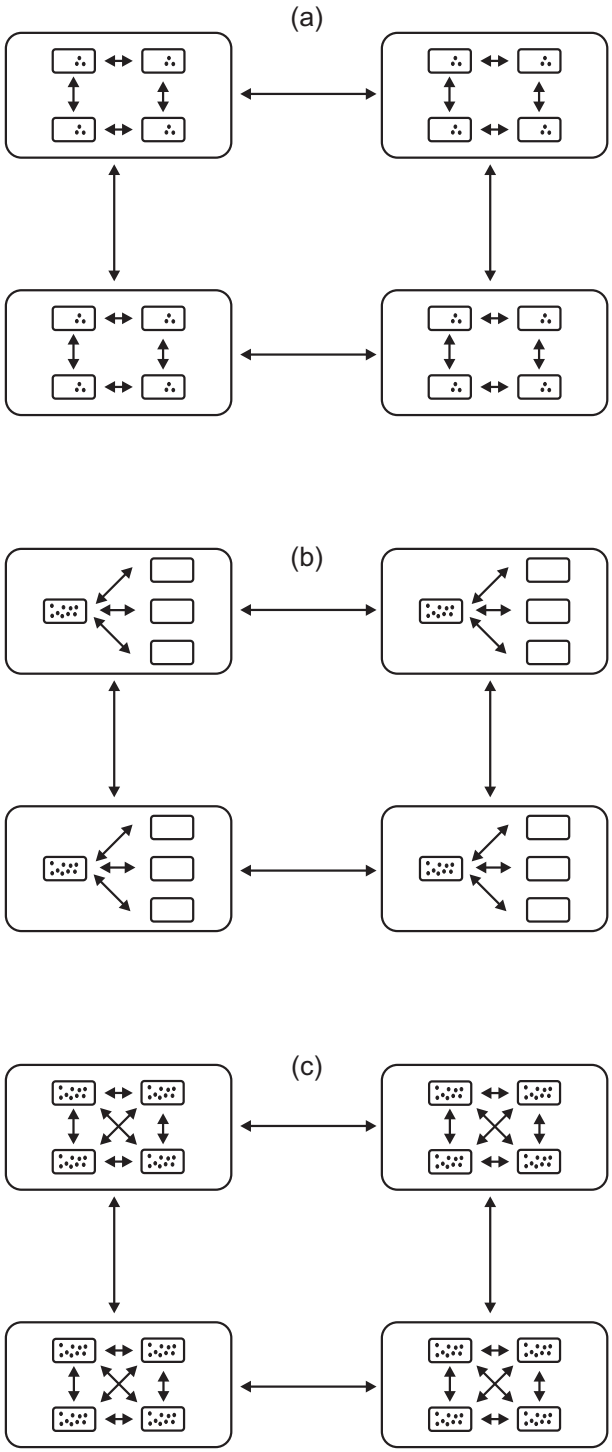

Figura 2.19: Arquitecturas híbridas con niveles alto grano grueso y nivel bajo (a) grano Fino, (b) maestro-esclavo, (c) grano grueso en arquitecturas de grano fino.

- **Métodos de ajuste basados en funciones**. Determinan el valor utilizando una fórmula que depende de la generación (t) en la que se encuentre el proceso de búsqueda.

2.9.1.1. Ajuste constante

Este método mantiene constantes los valores de los parámetros, de ahí que sea el método de ajuste más simple. El método, se basa en el uso de una serie de reglas determinadas de forma empírica que, fijan el valor adecuado de cada parámetro de forma independiente. Aunque las reglas presentan una componente genérica importante, no son extrapolables a todos los EAs, siendo muy importante el tipo de codificación utilizada (binaria o real).

Los trabajos pioneros dentro de este ámbito e interesantes al mismo tiempo son los de De Jong [93], Grefenstette [69] y Schaffer [143]. Algunas de sus aportaciones más interesantes para GAs con codificación binaria fueron:

- De Jong propuso una $p_m = 0.001$, Grefenstette una $p_m = 0.01$ y que $p_m > 0.05$ tiende a ser perjudicial. Schaffer en la misma línea plantea una $p_m \in [0.005 \dots 0.01]$ corroborando la importancia del operador de mutación.

- En relación a la probabilidad de cruce, De Jong sugiere que no sea muy elevada, para evitar que de una generación a la siguiente la población no sea excesivamente diferente. Grefenstette, por contra, propone una $p_c = 0.95$ utilizando, eso sí, un método de selección elitista. Declara que, a medida que el tamaño de la población aumenta, la p_c debe decrementarse. Igualmente, Schaffer propuso $p_c \in [0.75 \dots 0.95]$ concluyendo que el GA es relativamente insensible a p_c y que a medida que se aumenta el tamaño de la población, su efecto parece diluirse.

- De Jong determinó que aumentar el tamaño de la población reduce los efectos estocásticos del muestreo aleatorio aunque, mejora el resultado del GA a largo plazo a costa de una respuesta inicial más lenta. Unos valores adecuados son $N \in [50 \dots 100]$. Greffenstette propuso unos límites similares $N \in [60 \dots 110]$ y Schaffer $N \in [20 \dots 30]$.

Para codificación real, existen menos trabajos al respecto aunque los rangos que se consideran adecuados son $p_m \in [0.6 \dots 0.7]$ y $p_c \in [0.7 \dots 1.0]$ según R. Ursem [155]. Particularmente, el autor de esta tesis prefiere utilizar valores de p_m menores $p_m \in [0.1 \dots 0.4]$ por los buenos resultados que éstos han presentado en diferentes aplicaciones [71].

2.9.1.2. Ajuste basado en funciones

El objetivo de este método es el de sustituir el parámetro constante p_x por una función $p_x(t)$, donde t representa la generación o iteración. La idea sería que el EA realizase inicialmente una exploración global y burda en las generaciones iniciales y una exploración

más local y precisa en las últimas. De ahí que, normalmente, este método focalice sus esfuerzos sobre el operador de mutación[20].

En **codificación binaria** lo habitual es modificar la probabilidad de mutación p_m de cada bit de manera que ésta decrezca a media que t aumente. Algunas propuestas son:

- Fogarty [53] propuso, inicialmente, una $p_m(t)$ exponencial decreciente

$$p_m(t) = \alpha + \frac{\beta}{2^t},$$

siendo α y β constantes. Posteriormente, amplió la expresión anterior para que la p_m fuese también dependiente de los bits del cromosoma. La idea es que los bits con mayor peso en la codificación tuviesen menor probabilidad de mutación y los de menor peso mayor

$$p_m(t,i) = \frac{\alpha}{2^{i-1}} + \frac{\beta}{2^{t+i-1}},$$

siendo α y β constantes e i la posición del bit dentro del cromosoma. Por ejemplo, si $L = 8$, $i = 1$ representaría el bit de menor peso e $i = 8$ el de mayor.

- Hesser y Männer sugirieron la siguiente expresión

$$p_m(t) = \sqrt{\frac{\alpha}{\beta}} \cdot \frac{e^{-0.5\gamma t}}{N\sqrt{L}},$$

donde α, β y γ son constantes, N el tamaño de la población y L el número de bits del cromosoma.

- Otra alternativa es la propuesta por Bäck y Schütz [6] donde:

$$p_m(t) = \begin{cases} \frac{T}{2T+(L-2)t} & 0 \leq t \leq T \\ \frac{1}{T} & T < t \end{cases}.$$

De esta manera, se consigue una probabilidad de mutación decreciente desde $p_m = 0.5$, cuando $t = 0$, hasta un $p_m = 1/L$ cuando $t = T$, manteniéndose constante a partir de ese momento.

- Janikow y Michalewicz [89] sugieren el uso del operador de mutación no uniforme para codificación binaria como una adaptación del utilizado en codificación real. El procedimiento determina el bit S_i de la codificación que tiene que conmutar:

$$i = \begin{cases} \lfloor \Delta(t) \rfloor & g = 0 \\ \lceil \Delta(t) \rceil & g = 1 \end{cases}$$

$$\Delta(t) = L\left(1 - r^{(1-t/T)^b}\right),$$

[20]Si se utiliza cruce por recombinación lineal o intermedia extendidas (ver sección 2.6.1.2) se pueden obtener resultados semejantes variando el parámetro d en función de la generación.

donde $r \sim U(0,1)$, T es el número máximo de generaciones, L el número de bits de la codificación binaria y b una constante. i es el índice del bit S_i a mutar que se obtiene del redondeo de $\Delta(t)$ a un número entero (por abajo $\lfloor\cdot\rfloor$ o por arriba $\lceil\cdot\rceil$) en función del número binario aleatorio g.

En **Codificación real** se suele actuar sobre la varianza del operador de mutación aleatoria con distribución gaussiana, haciéndolo dependiente de la generación t.

$$\theta_i^h = \theta_i^p + N(0, \alpha_i(t)).$$

Algunas alternativas son:

- Utilizar una función decreciente lineal

  $$\alpha(t) = \alpha_{ini} - \frac{\alpha_{ini} - \alpha_{fin}}{T} t,$$

  siendo T el número de generaciones y α_{ini} y α_{fin} los valores inicial $t = 0$ y final $t = T$ de α. Para que la recta sea de pendiente negativa $\alpha_{fin} < \alpha_{ini}$.

- Utilizar una función exponencial decreciente, como en la técnica *simulated annealing*

  $$\alpha(t) = \frac{1}{\sqrt{1+t}}.$$

- Métodos determinísticos no monótonos. Por ejemplo

  $$\alpha(t) = \beta\,(1 + sin(wt))$$

  con β y w constantes.

- Krink [100] propone el uso del operador SOC (*self-organized criticality*) donde $\alpha(t)$ tiene una componente estocástica, intentando que tome, generalmente, valores pequeños para realizar una exploración local y esporádicamente valores grandes que provoquen la exploración de zonas más lejanas.

Otro ejemplo, utilizado en codificación real es el operador de mutación no uniforme [120]. La mutación se realiza mediante las siguientes fórmulas:

$$\theta_i^h = \begin{cases} \theta_i^p + \Delta(t, \theta_{i\,max} - \theta_i^p) & g = 0 \\ \theta_i^p - \Delta(t, \theta_i^p - \theta_{i\,min}) & g = 1 \end{cases} \quad i \in [1 \ldots L]$$

$$\Delta(t, y) = y \cdot r \cdot \left(1 - \frac{t}{T}\right)^b,$$

donde la función $\Delta(t, y)$ devuelve un valor $\in [0 \ldots y]$, teniendo más probabilidades de obtener valores cercanos a cero a medida que $t \to T$. T es el número de generaciones, b es un parámetro, $r \sim U(0,1)$ y g un número binario aleatorio.

2.9.2. Métodos de ajuste adaptativos

Dentro de estos métodos los más utilizados son:

- **Métodos basados en medidas** de ciertas propiedades de la población en cada momento.
- **Métodos auto-adaptativos** donde los propios parámetros son codificados en el cromosoma del individuo.

2.9.2.1. Basados en medidas

Este método plantea la adaptación constante de los parámetros del EA, utilizando una serie de reglas función del estado de la población, el cual se determina, a través de medidas. Éstas pueden ser de dos tipos diferentes, medidas generacionales: tienen en cuenta la situación de la población en varias generaciones o instantáneas: sólo tienen en cuenta el estado de la población en la generación actual. Normalmente, los EAs utilizarán ambos tipos de medidas.

Algunos ejemplos de medidas generacionales serían:

- **Estancamiento**. El número de generaciones en las que no se ha mejorado el valor de la función de *fitness*.
- **Velocidad de convergencia**. La mejora de valor de *fitness* en un determinado número de generaciones.
- **Ratio de éxito de un operador**. El ratio entre el número de buenos individuos y el total de individuos producidos por un determinado operador genético.

En cuanto a las medidas instantáneas, la mayoría hacen hincapié en la medida de la diversidad de la población en el espacio de soluciones (θ)[21]. Algunas posibilidades son:

- **Diversidad genética**. Mide la similitud del genotipo de los individuos[22]. En el caso de espacios de búsqueda numéricos, una posibilidad sería determinar la distancia media entre cada individuo y
  - el resto de la población.
  - el individuo medio de la población.
  - el mejor de la población (mejor valor para función de *fitness*).
  - su individuo más cercano.

[21] Si el EA es un GP, el espacio utilizado para medir la diversidad es el de la función de *fitness*.

[22] En codificación binaria se suele utilizar la distancia de Hamming, mientras que en codificación real la distancia Euclídea.

- **Diversidad en el *fitness***. Mide la similitud del valor de *fitness* producido por los individuos.
- **Ratio en el *fitness***. Mide el ratio entre el mejor valor de *fitness* y
  - el valor de *fitness* medio.
  - el peor valor de *fitness*.

A continuación, se muestran algunos ejemplos de reglas utilizadas para la modificación de los parámetros que hacen uso de las medidas comentadas.

- **Hipermutación ante estancamiento**. Se trata de utilizar la medida generacional de estancamiento, comentada anteriormente, para mutar todos los individuos, con una varianza elevada, cuando el número de generaciones con estancamiento sea superior a un determinado valor preestablecido.
- **Fin de la búsqueda**. Normalmente, las medidas instantáneas de diversidad suelen utilizarse como criterios de terminación del propio EA. Por ejemplo, si la medida de la diversidad es inferior a un determinado valor el algoritmo se termina.
- **Regla de mutación gaussiana de Rechenberg** utilizada en los (1+1)-ES. Aplica la medida de ratio de éxito p_s del operador de mutación con la relación 1/5 [8]. La actualización de $\alpha(t)$ se realiza cada n generaciones, de manera que, si el $p_s > 1/5$, la varianza debería incrementarse $\alpha(t) = \alpha(t-n)/c$ y si $p_s < 1/5$ debería decrementarse $\alpha(t) = \alpha(t-n)/c$. El parámetro c está en el rango $[0.817 \ldots 1.0]$.
- **Reglas difusas**. Algunas reglas están fundamentadas en sistemas de control difuso, utilizan diferentes medidas para controlar parámetros como N, p_c y p_m. El problema es que es necesario implementar el motor de inferencia, lo cual no es nada trivial.
- **Regla de envejecimiento**. El algoritmo GAVaPS [120] (*Genetic Algorithm with varing population size*) modifica el parámetro N (tamaño de la población) utilizando una medida generacional, que indica como de viejos son los individuos. Cuando su edad (medida en número de generaciones) supera su tiempo de vida son eliminados.

2.9.2.2. Auto-adaptativos

En este tipo de métodos, la idea pasa por introducir los parámetros del EA en el cromosoma. De esta manera, el cromosoma estará compuesto por los parámetros de la función de *fitness* θ y los propios parámetros del EA que se desea auto-adaptar.

$$\underbrace{\theta_1 \ldots \theta_L}_{\theta} \quad \underbrace{p_c, p_m, \ldots}_{\text{parámetros\_EA}} \quad .$$

Normalmente, se aplica el algoritmo que actualiza los parámetros del EA, y a continuación, con los nuevos parámetros, se actualiza θ. Se presupone que buenos individuos conllevarán buenos parámetros del EA. De manera que, al mismo tiempo que se localiza los parámetros adecuados del EA, se resuelve el problema de optimización.

Este tipo de métodos han sido y están siendo aplicados en estrategias evolutivas y programación evolutiva. El método más común es el operador de mutación gaussiana autoadaptativo sugerido por Schwefel [7]. El cromosoma está compuesto por θ y la varianza α

$$\underbrace{\theta_1 \dots \theta_L}_{\theta} \underbrace{\alpha_1 \dots \alpha_L}_{\alpha}$$

y el procedimiento de actualización de α y θ es el siguiente:

- Primero se actualiza α
$$\alpha_i^h = \alpha_i^p e^{N(0,\tau)} \;\; i \in 1 \dots L.$$
- Después se determina el θ del hijo
$$\theta_i^h = \theta_i^p + N(0, \alpha_i^h) \;\; i \in 1 \dots L.$$

2.10. Manejo de restricciones

El manejo de restricciones, dentro del ámbito de la optimización, es un tema ineludible, ya que todos los problemas reales estarán sometidos a restricciones, de una u otra manera. Las más evidentes, las que delimitan el propio espacio de búsqueda $D \in \mathcal{R}^L$. La incorporación de restricciones adicionales divide el espacio de búsqueda en dos regiones: la factible $\mathcal{F} \subseteq D$ donde, se satisfacen todas las restricciones, y la no factible $\mathcal{NF}$, donde, al menos, hay una restricción que no se satisface. Normalmente, no se suele asumir ninguna característica sobre estas regiones, de hecho no tienen porqué ser convexas e incluso podrían ser discontinuas (ver figura 2.20).

Formalmente, el problema de optimización con restricciones se puede generalizar de la siguiente manera.

$$\min_{\theta \in D} J(\theta) \tag{2.6}$$

sujeto a:

$$g_i(\theta) \leq 0, \; i = 1, \dots, m \tag{2.7}$$

$$h_j(\theta) = 0, \; j = 1, \dots, r \tag{2.8}$$

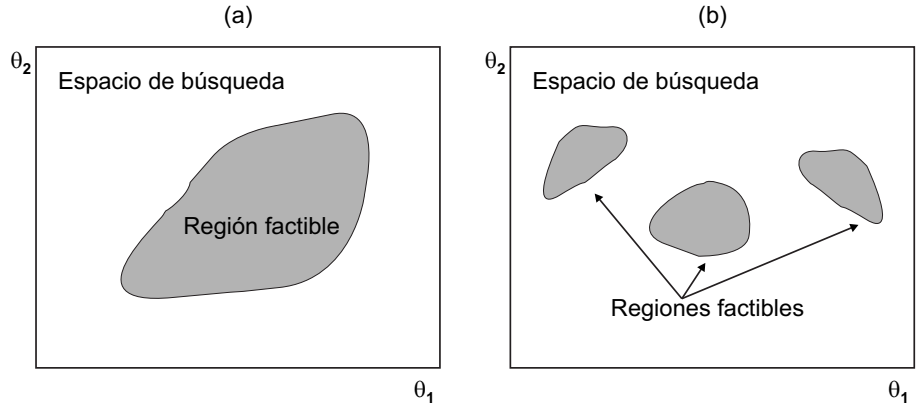

Figura 2.20: Espacio de búsqueda y su región factible. (a) $\mathcal{F}$ continua. (b) $\mathcal{F}$ discontinua.

donde θ es el vector de soluciones $\theta = [\theta_1, \ldots, \theta_L]$, m es el número de restricciones de tipo desigualdad y r de igualdad[23]. Por supuesto, tanto J como las restricciones h_j, g_i pueden ser no lineales.

Los algoritmos evolutivos, de por sí, son algoritmos de optimización sin restricciones[24], de ahí que sea necesario utilizar técnicas específicas para incorporarlas.

Algunas de las técnicas desarrolladas para manejar restricciones en los EAs son (ver [32, 122, 119] y sus referencias):

- **Funciones de penalización**. Es la forma más clásica de incorporar las restricciones. Para ello se penalizan las soluciones no factibles sobre la función de *fitness*.
- **Algoritmos de reparación**. Convierten las soluciones no factibles en otras soluciones factibles mediante la aplicación de ciertos procedimientos.
- **Codificaciones y operadores especiales**. El objetivo de esta técnica es mantener una población de soluciones factibles utilizando codificaciones específicas y operadores que imposibiliten la creación de soluciones no factibles.
- **Separación de objetivo y restricciones**. A diferencia de las funciones de penalización, lo que se pretende en este tipo de técnicas es tratar de forma separada la función de *fitness* y las propias restricciones.
- **Híbridas**. Proponen la interacción entre un EA y otra técnica que permita el manejo de restricciones.

A continuación, se procede a analizar con mayor detalle cada una de estas técnicas y sus posibles variantes.

[23]Generalmente, las restricciones de tipo igualdad (2.8) suelen transformarse en restricciones de desigualdad, de la siguiente manera: $|h_j(\theta)| - \epsilon_j \leq 0$, donde ϵ_j es un valor pequeño que indica la tolerancia permitida.

[24]A excepción de las restricciones que delimitan el propio espacio de búsqueda.

2.10.1. Funciones de penalización

La idea de esta técnica es transformar el problema de optimización con restricciones, en un problema de optimización sin restricciones incorporando un término adicional a la función de *fitness* que, dependa del conjunto de restricciones que no son satisfechas por una determinada solución.

De manera que[25]:

$$J'(\theta) = J(\theta) \pm \sum_{i=1}^{m} c_i G_i(\theta), \tag{2.9}$$

donde $J'(\theta)$ es la nueva función de *fitness* y c_i los coeficientes (positivos) de penalización de las restricciones. G_i son funciones dependientes de las propias restricciones, generalmente su relación será la siguiente:

$$G_i(\theta) = max[0, g_i(\theta)]^{w_i}, \tag{2.10}$$

siendo w_i parámetros constantes, normalmente iguales a 1 ó 2.

El principal problema de esta técnica reside en la determinación de los coeficientes de penalización c_i. Una elección inadecuada de los mismos podría dificultar la búsqueda a realizar por el EA. Valores altos en los coeficientes de penalización, provocarán que el EA deje de explorar la región no factible rápidamente, por lo tanto, si la solución óptima está en los límites (o cerca) de la región factible podría no ser localizada, ya que, la búsqueda del EA sería dirigida hacia el interior de la región factible. Por contra, si se escogen valores bajos para los coeficientes de penalización, se dedicará mucho tiempo a explorar la región no factible llegando a no distinguirse ésta de la factible y ralentizando el proceso de búsqueda.

Es posible penalizar los individuos no factibles de diferentes formas, por ejemplo:

- Sin tener en cuenta ninguna información sobre su proximidad a la región factible.
- Teniendo en cuenta información de la infactibilidad del individuo.
- Teniendo en cuenta el esfuerzo necesario que habría que realizar para convertir el individuo no factible en factible.

Evidentemente, cada alternativa plantea sus ventajas e inconvenientes. Algunas variantes para las funciones de penalización son las siguientes:

- Pena de muerte.
- Estáticas.
- Dinámicas.

[25]El signo ± será positivo en el caso de un problema de minimización y negativo para maximización.

2.10.1.1. Pena de muerte

Esta alternativa, hace desaparecer aquellos individuos que no cumplen las restricciones (de ahí su nombre) asignándoles un nivel de aptitud nulo. Esto simplifica el proceso de tratamiento de las soluciones no factibles, ya que no hay que elegir los coeficientes de penalización ($c_i = \infty$). Simplemente, se asigna un valor elevado a estas soluciones para que no sean seleccionadas[26] y poco a poco desaparezcan de la población. Las soluciones factibles son tratadas mediante la expresión genérica (2.9) y dado que $G_i(\theta) = 0 \ \forall i$, $J'(\theta) = J(\theta)$.

La simplicidad de esta técnica acarrea varios inconvenientes:

- Podrá funcionar adecuadamente si la región factible ocupa gran parte del espacio de búsqueda y es convexa. En caso contrario será ineficiente.
- No resultará adecuada en problemas con restricciones de igualdad.
- En situaciones donde todos los individuos de la población inicial no cumplan las restricciones[27] se producirá el estancamiento del EA ya que, todos los individuos recibirán el mismo nivel de aptitud.

2.10.1.2. Estáticas

Bajo esta alternativa, se agrupan aquellos enfoques en los que los coeficientes de penalización (c_i) no dependen de la generación (t) en la que se encuentre el proceso de búsqueda, es decir, permanecen constantes. Existen muchas propuestas, de diferentes autores, para escoger la forma de penalizar las restricciones, algunas de ellas se comentan a continuación.

- En [84] se propone escoger los coeficientes $c_i = c(q_i)$, donde $c(q_i)$ es una función que determina el valor de c_i, según el nivel de violación q_i de la restricción. La idea es balancear las restricciones de forma separada. El problema que plantea este método está relacionado con la definición de los niveles de violación y la función $c(q_i)$ que aunque hace la técnica muy flexible, dificulta su utilización por la gran cantidad de parámetros a fijar.
- Una alternativa más sencilla es la planteada en [101] donde:

$$J'(\theta) = \begin{cases} J(\theta), & \theta \in \mathcal{F} \\ K - \sum_{i=1}^{s} \frac{K}{m}, & \theta \in \mathcal{NF} \end{cases},$$

siendo m el número total de restricciones, s las restricciones no satisfechas y K una constante elevada. Con este procedimiento, los individuos $\theta \in \mathcal{NF}$ no necesitan ser

[26] El equivalente a utilizar coeficientes $c_i = \infty$.

[27] Situación nada atípica, si se tiene en cuenta que la población inicial se crea, generalmente, de forma aleatoria y que la región factible podría ocupar un pequeña porción del espacio de búsqueda.

evaluados. Por otra parte, dos individuos que violen el mismo número de restricciones tendrán el mismo valor de *fitness*. Esto hace que el método, aunque sea sencillo de aplicar, presente problemas especialmente en casos donde el espacio $\mathcal{NF}$ constituya gran parte de D.

- Otra alternativa es la planteada en [82] para la que:

$$J'(\theta) = J(\theta) \pm \sqrt{\sum_{i=1}^{m} H(-g_i(\theta))g_i(\theta)^2},$$

donde:

$$H(y) = \begin{cases} 1, & y > 0 \\ 0, & y \leq 0 \end{cases}.$$

La ventaja de esta técnica, al igual que la anterior es que no hay que elegir ningún coeficiente de penalización. Por otra parte, $J(\theta)$ no depende sólo del número de restricciones no satisfechas, también depende del grado de insatisfacción de las mismas $g_i(\theta)$.

2.10.1.3. Dinámicas

Bajo esta categoría se aglutinan aquellas técnicas donde los coeficiente de penalización dependen de la generación (t). La inclusión de la variable generación (t) tiene por objetivo que las restricciones sean poco penalizadas en la generaciones iniciales, asegurando una buena exploración del espacio de búsqueda. A medida que el proceso evoluciona, el grado de penalización de éstas aumenta favoreciendo así la supervivencia de las soluciones factibles.

Una propuesta en [91] plantea la utilización de los siguientes coeficientes de ponderación:

$$c_i = (C \cdot t)^{\alpha}, \tag{2.11}$$

donde C y α son constantes (los valores sugeridos por los autores son $C = 0.5$ y $\alpha = 1$). Como se puede deducir de (2.11), a medida que aumenta el número de generaciones, se incrementa la penalización de las restricciones.

Dentro de esta categoría de penalizaciones dinámicas existen dos enfoques específicos:

- Penalización *Simulated Annealing*.
- Penalización Adaptativa.

En la penalización ***Simulated Annealing*** los coeficientes de penalización van variando simulando el proceso de enfriamiento característico de la técnica *Simulated Annealing*.

En [121] se propone la modificación de los coeficientes c_i, cada cierto número de generaciones, de la siguiente forma:

$$c_i = \frac{1}{2\tau},$$

donde τ representa el horario de enfriamiento $\tau = \tau_o - C \cdot t$, τ_o y C son parámetros constantes. El proceso se detiene cuando se alcanza un valor final τ_f.

En la penalización ***Adaptativa***, se modifican los coeficientes de penalización teniendo en cuenta, la generación (t) y, además, la información adicional de la población (*fitness*, factibilidad, etc.). En [16] por ejemplo, se actualizan los coeficientes de penalización de la siguiente manera:

$$c_i = \lambda(t),$$

$$\lambda(t) = \begin{cases} \frac{1}{\beta_1}\lambda(t), & \text{caso A} \\ \beta_2\lambda(t), & \text{caso B} \\ \lambda(t), & \text{caso distinto de A o B} \end{cases},$$

donde $\beta_1, \beta_2 > 1$ y $\beta_1 \neq \beta_2$. El caso A, se da cuando el mejor individuo de las últimas k generaciones es factible y el caso B, cuando éste no fue nunca factible. La técnica necesita establecer un valor inicial $\lambda(0)$.

Es evidente que la penalización dinámica ofrece más posibilidades que la estática, sin embargo, siguen existiendo los mismos problemas de determinación de los diferentes coeficientes.

2.10.2. Algoritmos de reparación

Este enfoque plantea la idea de modificar (reparar) una solución no factible $\theta_{\mathcal{NF}i} \in \mathcal{NF}$, de manera que su versión modificada $\theta_{\mathcal{F}i}$ pertenezca a la región factible, $\theta_{\mathcal{F}i} \in \mathcal{F}$. Esta tarea tiene dos objetivos:

- Poder evaluar, adecuadamente, la solución factible mediante la función de *fitness*, $J(\theta_{\mathcal{F}i})$.
- Posibilitar la sustitución de la solución no factible por la factible en la población $P(t)$, $\theta_{\mathcal{NF}i} \Rightarrow \theta_{\mathcal{F}i}$.

El procedimiento para reparar una solución no factible es dependiente del problema a tratar. En ocasiones, reparar una solución no factible podría llegar a ser tan complejo como resolver el problema original.

Otras veces, por suerte, el procedimiento es más sencillo, como por ejemplo en optimización combinatoria. En este caso el procedimiento de reparación de las soluciones no factibles mejorará las prestaciones del EA.

Algunos procedimientos transforman las soluciones no factibles aplicando sobre ellas una búsqueda local que lleve la solución hasta la región factible (por ejemplo, cruzando la solución no factible con una factible como en GENOCOP III [123]).

Una vez reparada la solución, otro aspecto interesante a tratar es si la solución factible debe sustituir a la no factible en $P(t)$. Existen varias posibilidades:

- No reemplazar la solución no factible.
- Reemplazar siempre la solución no factible por la factible.
- Reemplazar determinado porcentaje de las soluciones no factibles por las factibles. En GENOCOP III, por ejemplo se reemplaza un 15 %, mientras que en [134] se propone un porcentaje de reemplazo del 5 %.

2.10.3. Codificaciones y operadores especiales

La utilización de codificaciones y operadores especiales tiene como objetivo principal generar y preservar soluciones factibles. El cambio de codificación tiene por objetivo la simplificación del espacio de búsqueda, mientras que los operadores (modificados debido al cambio de representación) son los encargados de mantener la factibilidad de los nuevos individuos que se vayan generando.

Por lo general, si se dispone de la codificación y operadores adecuados para un problema particular, esta técnica resultará muy eficiente (sin duda más que la técnica de ponderación), sin embargo su generalización, a otro tipo de problemas, seguramente no será evidente.

La particularidad de esta técnica ha provocado la aparición de muchos enfoques o variantes de aplicación de la misma, cada uno con sus ventajas e inconvenientes [39]. Uno de los enfoques más genérico es el basado en el uso de decodificadores.

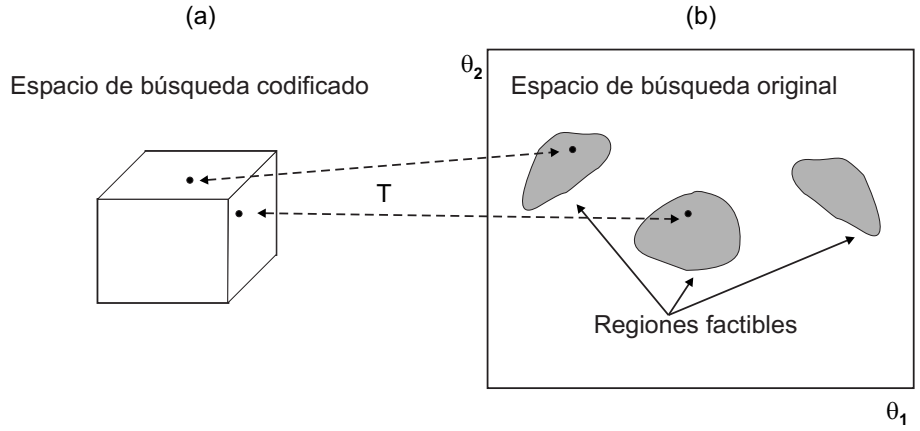

Figura 2.21: Relación entre espacio original y decodificado, a través del decodificador T.

En este enfoque la codificación del individuo no representa directamente una solución al problema, sino que suministra las instrucciones para construirla (decodificarla)[28]. Cuando se utilizan decodificadores, éstos imponen una relación entre las soluciones factibles $\theta \in \mathcal{F}$ y el espacio codificado (ver figura 2.21). Esta relación debe cumplir una serie de características fundamentales [122]:

- Para cada solución factible $\theta \in \mathcal{F}$ existe una solución codificada.
- Cada solución codificada corresponde a una solución factible.
- El proceso de codificación decodificación debe ser rápido desde el punto de vista computacional[29] y, generar pequeños cambios en la solución del espacio original cuando se producen pequeños cambios en la solución del espacio codificado.

2.10.4. Separación de objetivo y restricciones

Dentro de esta categoría, donde la función de *fitness* no es combinada con el conjunto de restricciones es posible encontrar varios enfoques:

- Poblaciones coevolutivas.
- Superioridad de puntos factibles.
- Memoria conductista.
- Optimización multiobjetivo.

A continuación se procede a describir cada uno de estos enfoques.

2.10.4.1. Poblaciones coevolutivas

En [135] se plantea un enfoque donde hay dos poblaciones que evolucionan de forma conjunta. Una de las poblaciones contiene posibles soluciones al problema de optimización (factibles y no factibles) mientras que la otra población contiene las restricciones (no se trata exactamente de una población, ya que no evoluciona).

El *fitness* de la primera población representa el nivel de satisfacción de las restricciones, mientras que en la segunda el nivel de violación, cantidad de individuos que no satisfacen una determinada restricción. La presión de selección de los individuos de una población depende del *fitness* de los individuos de la otra población. La idea de este enfoque es incrementar el *fitness* de las restricciones que son más duras de satisfacer para que la búsqueda se concentre en ellas.

[28] El ejemplo más obvio es el de la decodificación binaria a real de la sección (2.4).

[29] Aunque este requisito no es indispensable desde el punto de vista funcional, sí que lo es desde el punto de vista de la eficiencia del mismo.

2.10.4.2. Superioridad de puntos factibles

La idea que se utiliza en este enfoque es que cualquier solución factible siempre será mejor que otra que no sea factible. Bajo esta premisa, algunos autores han planteado formas diferentes de incorporar el tratamiento de las restricciones a los EAs.

Por ejemplo en [136] los individuos son evaluados teniendo en cuenta el siguiente procedimiento:

$$J'(\theta) = \begin{cases} J(\theta), & \theta \in \mathcal{F} \\ 1 + \sum_{i=1}^{m} G_i(\theta), & \theta \in \mathcal{NF} \end{cases},$$

escalando $J(\theta)$ en el rango $(-\infty, 1)$ (para el caso de minimización) y como $G_i(\theta) > 0$ $\forall \theta \in D$, siempre se cumplirá que las soluciones factibles producirán un mejor valor de *fitness* que las no factibles.

Un planteamiento similar se muestra en [41] donde los individuos son evaluados de la siguiente manera:

$$J'(\theta) = \begin{cases} J(\theta), & \theta \in \mathcal{F} \\ J(\theta_{max}) + \sum_{i=1}^{m} G_i(\theta), & \theta \in \mathcal{NF} \end{cases},$$

siendo θ_{max} el individuo $\in \mathcal{F}$ que, presenta el mayor valor de *fitness* de todos los individuos factibles de la población $P(t)$. Si no existe ningún individuo factible, se fija $J(\theta_{max}) = 0$. Con este enfoque se evita tener que escalar $J(\theta)$. El método utiliza un procedimiento de selección basado en torneo (ver sección 2.5.2) donde compiten dos individuos teniendo en cuenta que siempre se prefiere una solución factible a una no factible. Un inconveniente de este método es la alta presión de selección que se ejerce, lo cual actúa en detrimento de la diversidad en la población, de ahí que sea necesario utilizar técnicas de nichos (ver 2.11.1.2).

2.10.4.3. Memoria conductista

La particularidad de esta técnica reside en que las restricciones son tratadas en un orden en particular y de forma secuencial utilizando el siguiente procedimiento [144]:

- Inicializar $P(t)$ con individuos aleatorios $\theta \in D$ y el contador de restricciones $i = 1$.
- Hacer evolucionar la población sobre la función $g_i(\theta)$, hasta que un porcentaje ϕ de la población sea factible teniendo en cuenta únicamente la restricción i.
- $i = i + 1$.
- Sin modificar la población, hacerla evolucionar de nuevo, igual que en el paso anterior, pero eliminando aquellos individuos que no satisfacen las restricciones $1, 2, \ldots, i - 1$.
- Si $i < m$ repetir los dos pasos anteriores y si $i = m$ optimizar la función $J(\theta)$ eliminando los individuos que no satisfacen las restricciones.

Evidentemente el procedimiento exige que las restricciones sean ordenadas, lo cual influye en el tiempo de ejecución y precisión del algoritmo. Por otra parte, la última fase del algoritmo corresponde exactamente al enfoque de pena de muerte del apartado 2.10.1.1, con la diferencia que se parte de una población donde todos los individuos son factibles.

2.10.4.4. Optimización multiobjetivo

Lo que se pretende en este enfoque es transformar las restricciones en funciones a optimizar, convirtiendo así el problema en uno de optimización multiobjetivo (ver capítulo 3). Se trata de optimizar los elementos del vector **J**.

$$\mathbf{J} = (J(\theta), G_1(\theta), G_2(\theta), \ldots, G_m(\theta)).$$

Una solución ideal θ_{ideal} será aquella que produzca $G_i(\theta_{ideal}) = 0, i \in [1, \ldots, m]$ y que $J(\theta_{ideal}) \leq J(\theta), \forall\theta \in \mathcal{F}$ (para el caso de minimización).

2.10.5. Híbridas

Bajo esta categoría se encuentran englobados todos aquellos algoritmos evolutivos que se han hibridado con otras técnicas, generalmente técnicas de optimización más adecuadas para el manejo de restricciones. La variedad y particularidad de las técnicas desarrolladas por diferentes autores son muchas (hibridación con método *simplex*, multiplicadores de Lagrange, lógica difusa, *simulated annealing*, etc.) como se muestra en [119].

Para que sirva de ejemplo se menciona el planteamiento mostrado en [1] donde presenta un híbrido entre un EA y los multiplicadores de Lagrange. La función de *fitness* que se plantea es la siguiente:

$$J'(\theta) = J(\theta) + \frac{1}{2}\sum_{i=1}^{m}\gamma_i(G_i(\theta) + \mu_i)^2,$$

donde $\gamma_i > 1$ y μ_i es un término asociado a la restricción i que se define teniendo en cuenta el máximo nivel de violación de la restricción escalándolo con un parámetro $\beta > 1$ definido por el usuario. Este parámetro β se utiliza también para incrementar γ_i en cada generación, con el objetivo de que el efecto de penalización crezca a medida que t aumente.

2.11. Optimización de funciones multimodales

Los Algoritmos Evolutivos están especialmente indicados para evitar los mínimos locales (presentes en las funciones multimodales) y converger hacia el óptimo global, debido principalmente a que realizan una búsqueda paralela con todos los individuos de la población a la vez.

Sin embargo, en ocasiones es deseable detectar, al mismo tiempo que el óptimo global, los diferentes óptimos locales, pues podrían resultar ser buenas soluciones, que tras ser estudiadas por un experto (mediante criterios adicionales) podrían ser utilizadas en un situación real. Incluso es posible encontrarse con un escenario donde existan varios óptimos globales, el ejemplo más evidente es aquel en el que se desea optimizar varias funciones al mismo tiempo (problemas de optimización multiobjetivo) y no existe una solución óptima para todas ellas (pueden existir infinitas soluciones óptimas conocidas como óptimos de Pareto[30]).

Otros casos donde interesa detectar varios óptimos pueden encontrarse en [147], relacionados con el aprendizaje de conocimiento en decisiones financiaras y en [85] donde se identifican reglas para sistemas de clasificación.

La clave para poder detectar diferentes óptimos pasa por asegurar la diversidad en la población evitando la convergencia de ésta hacía un sólo punto o zona dentro del espacio de búsqueda. Evidentemente, la diversidad irá en detrimento de la precisión del algoritmo a la hora de dar con las soluciones definitivas.

Existen muchos algoritmos dedicados a la optimización de funciones multimodales. Una posible clasificación de los mismos podría dividir estos en dos grupos [155]:

- **Algoritmos de evitación**. Aquellos que incorporan mecanismos que evitan la convergencia (concentración) de la población.
- **Algoritmos de reparación**. Estos algoritmos se encargan de reparar (desconcentrar) la población cuando se detecta la convergencia de la misma.

2.11.1. Algoritmos de evitación

Bajo esta categoría se encuentran la mayoría de técnicas existentes [111]. Para evitar la convergencia se utilizan mecanismos que reduzcan la velocidad de convergencia o que prevengan el hacinamiento de individuos en una misma zona. Algunos de los mecanismos más conocidos son:

- Hacinamiento (*Crowding*).
- Apareamiento restringido (*Mating restriction*).
- *Fitness* compartido (*Fitness sharing*).
- División del espacio de búsqueda (*Search space division*).

[30] Ver capítulo 3 de optimización multiobjetivo con algoritmos evolutivos.

2.11.1.1. Hacinamiento

El mecanismo de hacinamiento (*crowding mechanism*) [93] reduce la convergencia prematura minimizando los cambios en la población entre diferentes iteraciones. Para ello, lo que hace es crear un número G de descendientes a partir de la población $P(t)$. A continuación, se analiza uno por uno, si estos individuos deben incorporarse a la población (selección de estado uniforme, sección 2.5.3). Por esto, para cada individuo descendiente θ_o se selecciona (aleatoriamente) un conjunto de individuos CF (*crowding factor*) de la población $P(t)$ y, se compara el individuo θ_o con el individuo más similar[31] de CF, θ_s. Si θ_o es mejor (desde el punto de vista del valor de la función de *fitness*) que θ_s, sustituirá a éste en $P(t)$, de lo contrario θ_s permanecerá en la misma. De esta manera, se evita, en gran medida, la tendencia del EA de hacinar la población en un punto. Este mecanismo presenta algunos problemas como son:

- Retarda la exploración de zona del espacio de búsqueda que no estén cerca de individuos de la población inicial.
- Si el tamaño del conjunto CF es pequeño podría reemplazar una buena solución por otra, también buena, si en CF no hay individuos cercanos a θ_o (reemplazamiento con error).

Una modificación de este mecanismo, **hacinamiento determinístico** [110] se creó con la intención de eliminar los parámetros CF y G, reducir el número de reemplazamientos con error e incrementar la presión de selección entre individuos que comparten el mismo óptimo, favoreciendo la convergencia sobre éstos.

En el hacinamiento determinístico modifica el mecanismo de reemplazamiento anterior reduciendo el conjunto de individuos CF a contener los progenitores de θ_o. Lo cual tiene sentido, dado que los descendientes suelen parecerse a sus progenitores.

En [116] se presenta una variante del mecanismo anterior llamada hacinamiento probabilístico, en la que se sustituye el mecanismo de reemplazamiento determinístico por uno estocástico. La probabilidad de reemplazo (entre un progenitor θ_p y su descendiente θ_o) viene determinada por la siguiente expresión:

$$prob\_reemplazo = \frac{J(\theta_o)}{J(\theta_p) + J(\theta_o)}.$$

Esta variante presenta el problema de ser demasiado sensible los valores de la función de *fitness*, de ahí que la presión de selección es bastante dependiente del problema en cuestión.

2.11.1.2. *Fitness* compartido

Esta técnica se encarga de promover la formación y mantenimiento de subpoblaciones de $P(t)$ conocidas como nichos (ver estudio realizado en [111]). Las formación de nichos

[31] La distancia puede ser determinada en el dominio del genotipo o en el del fenotipo.

tiene como objetivo la localizar diferentes óptimos al mismo tiempo.

La idea es modificar el valor de la función de *fitness* de manera que ésta tenga en cuenta la densidad de población del nicho al que pertenece un determinado individuo. En [62] se planteó la primera función de *fitness* compartido $J_i^{sh}(\theta)$ para un individuo i como:

$$J_i^{sh}(\theta) = \frac{J_i(\theta)}{\sum_{j=1}^{N} sh(d(i,j))}, \tag{2.12}$$

donde N es el tamaño de la población, $J_i(\theta)$ la función de *fitness* original del individuo i y $sh(\cdot)$ es una función que refleja la densidad del nicho del individuo i y que se define como:

$$sh(d(i,j)) = \begin{cases} 1 - \left(\frac{d(i,j)}{\sigma_{sh}}\right)^{\alpha}, & d < \sigma_{sh} \\ 0, & \text{Otro caso} \end{cases},$$

donde $d(i,j)$ es la distancia[32] entre los individuos i y j, α es una constante normalmente con valor unitario y σ_{sh} (radio del nicho) es la distancia estimada entre nichos. En [44] se muestran una serie de fórmulas para determinar el valor de σ_{sh} en función del número de nichos esperados y del volumen del espacio de búsqueda.

Este procedimiento, tan popular, plantea una serie de inconvenientes:

- Determinar el número de nichos (posibles óptimos), en muchas ocasiones, no es posible y simplemente se reduce a una suposición.
- Calcular $J_i^{sh}(\theta)$ supone una carga computacional considerable, especialmente para poblaciones con un elevado número de individuos.
- La robustez de este método puede verse comprometida, dada su sensibilidad al rango de valores que puede tomar la función de *fitness*, como ha sido demostrado en [155].

2.11.1.3. Apareamiento restringido

En optimización de funciones multimodales el operador de cruce puede producir el fenómeno de ruptura[33], cuando se cruzan dos individuos pertenecientes a dos nichos, ya que el resultado podría ser un descendiente que no perteneciese a ninguno de los nichos a los que pertenecen sus progenitores.

Para evitar este problema, se puede restringir el apareamiento de modo que, sólo pueden cruzarse individuos que pertenezcan al mismo nicho, es decir, cercanos entre sí. Así

[32] Como casi siempre, la métrica utilizada puede ser la del genotipo (distancia Hamming) o la del fenotipo (métrica euclídea), aunque esta última es la que suele dar mejores resultados.

[33] Se produce cuando el operador, en este caso el de cruce, destruye información genética buena de los progenitores [111].

que, primero se selecciona un individuo de la población de forma aleatoria y a continuación, se busca un determinado individuo de la población cuya distancia al anterior sea inferior a un parámetro $\sigma_{restric}$ conocido como radio de apareamiento. Si no se puede encontrar ningún individuo dentro de este radio se escoge uno aleatoriamente.

Esta técnica también acarrea el problema del coste computacional que planteaba la técnica anterior de *fitness* compartido.

2.11.1.4. División del espacio de búsqueda

La idea que persiguen estos métodos es mantener múltiples subpoblaciones minimizando el solape entre ellas. Estas subpoblaciones cubrirían diferentes áreas del espacio de búsqueda (que se vería dividido) facilitando la detección de diferentes óptimos en paralelo. De esta manera, la utilización del operador de cruce entre individuos de la misma subpoblación, evitaría el fenómeno de ruptura comentado en el apartado anterior.

Aunque existen varios algoritmos englobados bajo esta categoría [155] sólo se comentará, para que sirva de ejemplo, el algoritmo evolutivo multinacional (MEA) presentado en [153].

MEA divide la población $P(t)$ en un tipo de subpoblaciones llamadas naciones $N_k(t)$, cada una encargada de localizar un óptimo. Una nación está constituida por una conjunto de individuos, un gobierno y una política. El gobierno es un subconjunto de individuos de la nación, elegidos por ser los más representativos de ésta (mejor valor de la función de *fitness*). A partir del gobierno es calculado la política, que simplemente será un punto dentro del espacio de búsqueda que representará el óptimo a alcanzar (ver figura 2.22).

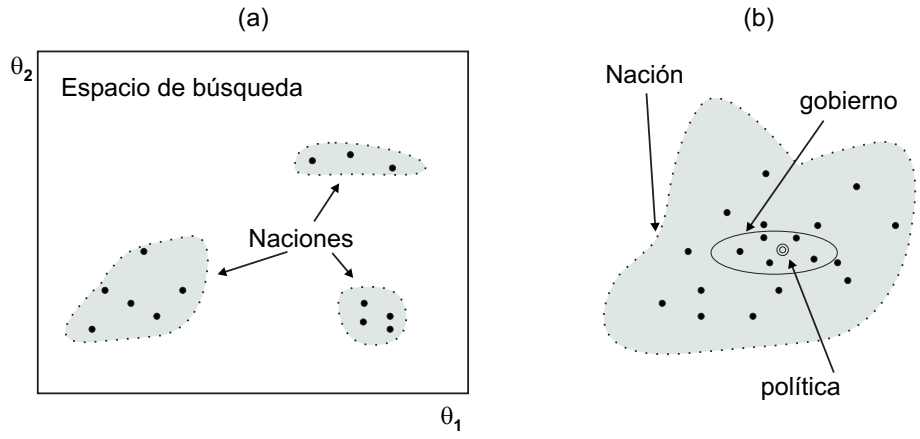

Figura 2.22: (a) Espacio de búsqueda con tres naciones. (b) Nación con su gobierno y la política calculada.

Asociado a la nación existen una serie de reglas:

- Migración. Se encarga de mover individuos entre naciones y crear nuevas naciones en áreas no pobladas ante la detección de valles y colinas. Para detectar un valle,

por ejemplo, se chequea si entre dos puntos (uno de ellos la política y el otro un individuo migrado) existen puntos intermedios cuyo *fitness* presenta un valor menor que los dos puntos en cuestión.

- Fusión. Se encarga de fusionar dos naciones que intentan alcanzar el mismo óptimo.
- Selección. Su función es la de seleccionar los individuos de la nación a cruzar.
- Elección. Determina como es escogido el gobierno de la nación. Una posibilidad es escoger un determinado porcentaje de los mejores (mejor valor de la función de *fitness*) individuos de la nación.

2.11.2. Algoritmos de reparación

Dentro de esta categoría, es posible, a su vez, discernir entre dos tipos diferentes de estrategias:

- Extinción de la masa (*Mass extinction*). Donde se mantiene la diversidad reemplazando parte de la población.
- Reinicialización y etapa (*Restart and phased*). Donde se introduce nuevo material genético cuando se detecta la convergencia de la población.

2.11.2.1. Extinción de la masa

Las diferentes técnicas de extinción de la masa se diferencian en cómo son seleccionados los individuos que van a ser eliminados (individuos que se van a extinguir) y cómo serán creados los nuevos individuos que sustituirán a éstos.

Una de las formas más sencillas de aplicar está técnica, consiste en reemplazar un porcentaje de individuos de la población (entre el 5 % y 10 % típicamente, seleccionados aleatoriamente) por nuevos individuos creados aleatoriamente [68].

En [67] el procedimiento para determinar si un individuo es reemplazado es el siguiente:

- Primero se genera un factor de estrés $\eta \sim U(0, 0.96)$[34].
- A continuación, se escala el *fitness* del individuo en el rango $[\alpha, 1]$ mediante la siguiente expresión:

$$J'(\theta_i) = \alpha + (1 - \alpha)\frac{J(\theta_i) - J(\theta_{max})}{J(\theta_{min}) - J(\theta_{max})},$$

donde α es un parámetro, $J(\theta_{max})$ y $J(\theta_{min})$ son los valores de la función de *fitness* del peor y mejor individuo respectivamente.

[34]El límite 0.96 está motivado porque el evento de mayor devastación sobre la tierra fue la extinción del 96 % de todos los animales marinos.

- Si $J'(\theta_i) < \eta$ el individuo es eliminado y su hueco es rellenado con mutaciones de los individuos restantes en la población.

En [12] el porcentaje de individuos a eliminar es determinado utilizando una distribución de ley de potencia y los huecos dejados por los individuos eliminados son rellenados por copias mutadas de éstos.

2.11.2.2. Reinicialización y etapa

Las técnicas que se aglutinan bajo esta categoría se caracterizan por detectar la convergencia de la población y *a posteriori* aplicar procedimientos para eliminarla. La optimización es abordada mediante una serie de etapas o fases, a que los mejores individuos sobreviven.

Por ejemplo en [50] se plantea el algoritmo CHC que utiliza una estrategia de reinicialización cuando detecta que la población a convergido, reinicializando la población con copias mutadas del mejor individuo. El operador de mutación sólo se utiliza en la fase de reinicialización, mientras que en la fase de optimización se utiliza el operador de cruce uniforme.

Otro ejemplo es el algoritmo DGEA [154] que alterna entre dos fases, una de exploración y otra de explotación. El algoritmo determina el grado de diversidad, mediante la medida de la distancia al punto medio de la población utilizando la siguiente expresión:

$$div(P) = \frac{1}{|D| \cdot |N|} \cdot \sum_{i=1}^{N} \sqrt{\sum_{j=1}^{L} (\theta_j^i - \bar{\theta}_j)^2},$$

donde $|D|$ es la longitud de la diagonal del espacio de búsqueda $D \in \mathcal{R}^L$, N el número de individuos de la población, θ_j^i es la dimensión j del individuo i y $\bar{\theta}$ la dimensión j del punto medio $\bar{\theta}$.

En la fase de exploración el algoritmo (al igual que el CHC) sólo aplica el operador de cruce y cuando detecta que la diversidad está por debajo de un valor límite $div_{min} > div(P)$ se conmuta a la fase de explotación, donde se utiliza el operador de mutación, hasta que $div(P) > div_{max}$ momento en el que se retorna a la fase de exploración.

2.12. Conclusiones

En este capítulo se han tratado los aspectos básicos de los EAs relacionados con la codificación, los diferentes operadores, la problemática de las funciones a optimizar y el ajuste de los parámetros propios de los EAs, de una forma genérica, sirviendo como punto de partida para poder describir, en capítulos posteriores, los algoritmos desarrollados en esta tesis.

Por otra parte, también se ha hecho bastante hincapié en aspectos más particulares como optimización de funciones con restricciones, funciones multimodales e implementación paralela de EAs. Estos aspectos serán de mucha utilidad para resolver los diferentes problemas que se irán planteando a lo largo de la tesis.

Capítulo 3

Algoritmos Evolutivos Multiobjetivo

3.1. Introducción

En el capítulo anterior se realizó un estudio de los algoritmos evolutivos como herramientas de optimización global de funciones de un único objetivo[1]. En este capítulo, se pretende mostrar el problema de la optimización de varios objetivos (funciones de *fitness*) simultáneamente y, mostrar las principales estrategias basadas en algoritmos evolutivos, conocidas como algoritmos evolutivos multiobjetivo (MOEA) [124].

Los problemas de optimización multiobjetivo (MOP) son bastante comunes en multitud de disciplinas, por ejemplo el diseño de sistemas complejos donde, por lo general, se desea minimizar los costes, maximizando las prestaciones. Normalmente, los objetivos a optimizar estarán en conflicto no existiendo una única solución ya que la solución que minimice los costes no maximizará las prestaciones y viceversa. Por lo tanto, existirá un conjunto de soluciones de compromiso, en un principio todas ellas igual de válidas, generalmente conocido como conjunto de **óptimos de Pareto**.

Encontrar todas las soluciones óptimas de Pareto no es trivial y supone un coste computacional elevado. De hecho, la optimización global de un MOP es un problema NP-completo[2] [8]. Además, dependiendo de los requerimientos del problema a resolver será necesario escoger una solución intermedia (entre coste y prestaciones) lo que supondrá todo un compromiso.

Por lo tanto, para resolver un MOP es necesario distinguir entre dos tipos de problemas, un problema de optimización (búsqueda del conjunto de óptimos de Pareto) y otro de determinación de la solución definitiva (*Decision making*). Dependiendo de como la fase de optimización y la de decisión se combinen, es posible clasificar las diferentes técnicas de resolución de MOP de la siguiente manera:

- **A priori**. Se toma la decisión de como seleccionar la solución antes que se realiza la búsqueda (decisión ⇒ búsqueda). Dado que la decisión se toma *a priori* se tiene la ventaja de que, a partir de ese momento, todos los esfuerzos se pueden concentrar en producir la solución final. Por contra, si la solución encontrada no es aceptable, el proceso debería repetirse hasta que se encuentre una apropiada. Evidentemente, si las preferencias entre los diferentes objetivos se modifican, es necesario repetir el proceso de búsqueda.

- **A posteriori**. Se realiza la búsqueda y, a continuación, se toma la decisión para seleccionar la solución (búsqueda ⇒ decisión). Este planteamiento está especialmente indicado cuando las preferencias sobre los objetivos no están claras o cambian, aunque los objetivos son los mismos, ya que no es necesario repetir el proceso de búsqueda. El inconveniente es que los costes computacionales pueden ser mayores que los de las técnicas *a priori*.

[1] Aunque se mencionó el concepto multiobjetivo fue para resolver un problema de un único objetivo con restricciones.

[2] Un problema es NP-completo si todos los algoritmos requeridos para resolverlo necesitan un tiempo exponencial, en el peor de los casos.

- **Progresivas**. La búsqueda y toma de decisiones se realiza de forma interactiva (búsqueda $\Longleftrightarrow$ decisión). Este método establece un compromiso entre los dos planteamientos anteriores, añadiendo los costes computaciones asociados a la interacción entre las etapas de búsqueda y decisión. Por otra parte, si la fase de decisión recae sobre un humano los costes en tiempo se verían seriamente agravados, aunque en un principio las decisiones tomadas serían mejores.

El uso de los EAs en la resolución de MOP se fundamenta principalmente en la posibilidad que éstos ofrecen de poder generar de forma paralela y en una única ejecución varios elementos del conjunto de óptimos de Pareto al mismo tiempo, gracias a la propia naturaleza poblacional de los EA. Numerosos autores han desarrollado diferentes operadores o estrategias para convertir los EAs originales en MOEAs que converjan hacia el conjunto de óptimos de Pareto con suficiente diversidad como para caracterizar a éste. Los buenos resultados mostrados por los MOEAs y la capacidad de éstos para abordar problemas de una gran variedad y complejidad han hecho que su uso sea cada vez mayor, siendo una de las ramas en auge dentro de los EAs [57, 168, 36, 2, 35].

El resto del capítulo se distribuye de la siguiente forma, la sección 3.2 muestra los conceptos básicos y la terminología que será utilizada a los largo del capítulo y en otros posteriores asociada a los MOPs. En la sección 3.3 se abordan los métodos clásicos de resolución de MOPs, se trata de técnicas que quedarían clasificadas como técnicas *a priori*. La sección 3.4 se centra de lleno en la descripción de los EAs más importantes utilizados en la resolución de MOPs. Estos MOEAs encajarían dentro del segundo grupo (*a posteriori*) donde se realiza una exploración del espacio de búsqueda para generar el conjunto de óptimos de Pareto (o una aproximación a éstos) y dejan en mano del diseñador la etapa de la selección de la solución más adecuada. La sección 3.5 trata los aspectos relacionados con la definición de métricas que permitan validar y analizar el funcionamiento de los MOEAs. Para finalizar la última sección se dedicará a las conclusiones del presente capítulo.

3.2. Conceptos y terminología de MOP

En esta sección se presentarán los conceptos básicos y la terminología asociada a los problemas de optimización multiobjetivo a través de una serie de definiciones.

Un MOP puede ser definido matemáticamente de la siguiente manera:

Definición 3.1 (MOP): *En general, un MOP minimiza*[3] *un vector de funciones* $\mathbf{J}(\theta) = [J_1(\theta), \ldots, J_s(\theta)] \in \Lambda \subseteq \mathcal{R}^s$*, respecto de un vector variable* $\theta \in D \subseteq \mathcal{R}^L$*.*

$$\min_{\theta \in D} \mathbf{J}(\theta).$$

$\triangle$

[3]Por simplicidad, todas las funciones $J_i(\theta)$ son convertidas a la forma de minimización teniendo en cuenta que $\max(J_i(\theta)) = -\min(-J_i(\theta))$.

Dicho de otra forma un MOP se define como el problema de encontrar el conjunto de vectores de variables de decisión que minimicen el vector de funciones cuyos elementos representan las funciones objetivo o de *fitness*.

La solución al MOP no es única (suponiendo que los objetivos, o al menos algunos de ellos están en conflicto) ya que la minimización de uno de ellos, puede suponer la maximización del otro y viceversa. Por ejemplo, la figura 3.1 muestra un caso particular de minimización de dos objetivos $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)]$ sobre el espacio de búsqueda $\theta = [\theta_1, \theta_2]$.

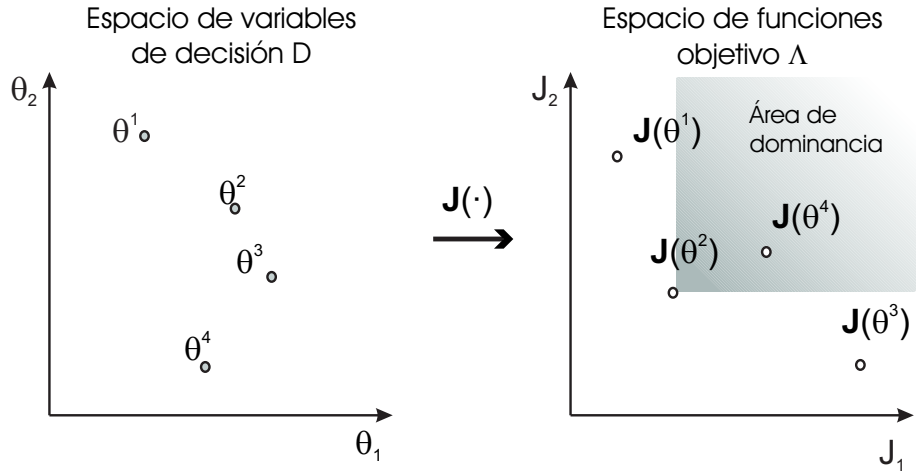

Figura 3.1: Ilustración del MOP de dos dimensiones. Concepto de Pareto dominante.

Los puntos θ^1, θ^2, θ^3 y θ^4 representan cuatro posibles soluciones al MOP. Del análisis de éstas soluciones (observando los valores de las funciones de *fitness* obtenidos con ellas) se podría determinar que la solución θ^1 es la mejor solución respecto de la función de *fitness* J_1 y la peor respecto de J_2 y justo lo contrario ocurriría con θ^3. Por lo tanto, *a priori* ninguna de las dos soluciones es mejor que la otra, y lo mismo ocurría con la solución θ^2, ya que ninguna de las tres mejora ambos objetivos (al mismo tiempo) respecto de las demás.

Sin embargo, la solución θ^4 es peor que la solución θ^2 (aunque no peor que θ^1 o θ^3) por lo tanto, a la hora de escoger una posible solución, habría que elegir entre θ^1, θ^2 o θ^3. Después del análisis gráfico del MOP ilustrado en la figura 3.1 se hace necesario expresar está situación matemáticamente definiendo el concepto de dominancia de Pareto para determinar si una solución domina a otra, es decir, es mejor.

Definición 3.2 (Pareto dominante): *Dado dos vectores θ^1, θ^2 en el espacio de soluciones, cuyos vectores en el espacio de las funciones de fitness son $\mathbf{J}(\theta^1)$ y $\mathbf{J}(\theta^2)$ respectivamente, se dirá que θ^1 domina a θ^2 (denotado como $\theta^1 \preceq \theta^2$) si y sólo si $\mathbf{J}(\theta^1)$ es parcialmente menor que $\mathbf{J}(\theta^2)$.*

$$\forall i \in A := [1, \ldots, s], J_i(\theta^1) \leq J_i(\theta^2) \wedge \exists i \in A : J_i(\theta^1) < J_i(\theta^2).$$

$\triangle$

Teniendo en cuenta la definición anterior se podría decir que $\theta^2 \preceq \theta^4$. Además, θ^2 dominaría a todo θ^j cuyo $\mathbf{J}(\theta^j) \in$ al área de dominancia marcada en gris en la figura 3.1.

Una vez definido el concepto de Pareto Dominante, se procede a definir lo que sería una solución óptima de Pareto al MOP y a partir de ésta, el conjunto de óptimos de Pareto Θ_P (solución al MOP) y el frente de Pareto $J(\Theta_P)$ que éste genera en el espacio de funciones de *fitness* (ver figura 3.2).

Definición 3.3 (Óptimo de Pareto): *Una solución θ^* es óptima de Pareto respecto de D, si y sólo si*

$$\nexists\, \theta \in D : \theta \preceq \theta^*.$$

$\triangle$

Definición 3.4 (Conjunto de óptimos de Pareto): *Dado un MOP $\mathbf{J}(\theta)$ se define el conjunto de óptimos de Pareto como:*

$$\Theta_P := \{\theta \in D |\, \nexists\, \theta' \in D : \theta' \preceq \theta\}.$$

$\triangle$

Definición 3.5 (Frente de Pareto): *Para un MOP $\mathbf{J}(\theta)$, cuyo conjunto de óptimos de Pareto es Θ_P, se define el frente de Pareto $\mathbf{J}(\Theta_P)$ como:*

$$\mathbf{J}(\Theta_P) := \{\mathbf{J}(\theta) |\, \theta \in \Theta_P\}.$$

$\triangle$

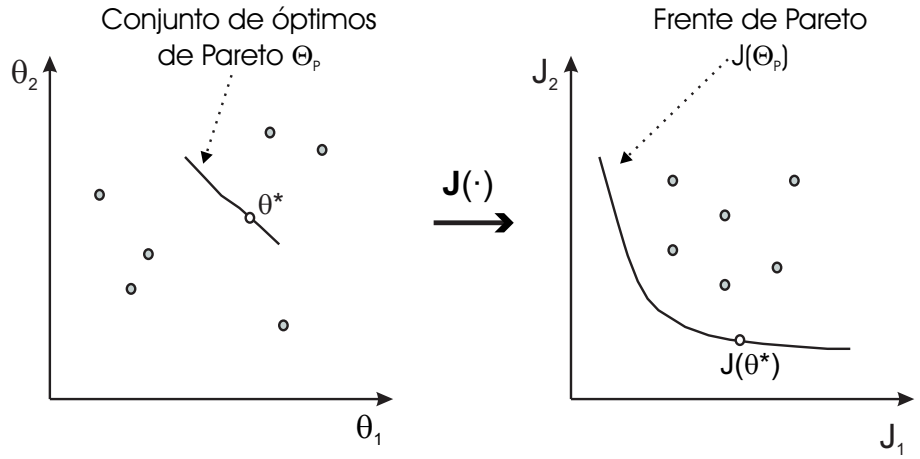

Figura 3.2: Conjunto de óptimos de Pareto Θ_P y el frente de Pareto que éstos producen $\mathbf{J}(\Theta_P)$ para un MOP de dos dimensiones. θ^* es óptimo de Pareto.

No es fácil encontrar una expresión matemática para la línea o hipersuperficie que forma el frente de Pareto, de hecho en la mayoría de los casos reales es imposible. La forma más evidente (aunque ineficiente) para determinar el frente de Pareto, pasa por

calcular $\mathbf{J}(D)$ y después seleccionar los puntos no dominados. En un principio, cuanto mayor sea el número de puntos calculados, mejor definido quedará el frente de Pareto.

En ocasiones, es suficiente (ya que determinar $\mathbf{J}(\Theta_P)$ es computacionalmente inabordable por ser, en la mayoría de los casos, un conjunto infinito de puntos $\in \Lambda$) con determinar un conjunto de puntos $\Theta_P^* \subset \Theta_P$, de manera que $\mathbf{J}(\Theta_P^*)$ caracterice adecuadamente $\mathbf{J}(\Theta_P)$ (ver figura 3.3).

Si bien Θ_P es un conjunto único, Θ_P^* no lo será, ya que para ello debería ser $\Theta_P^* = \Theta_P$.

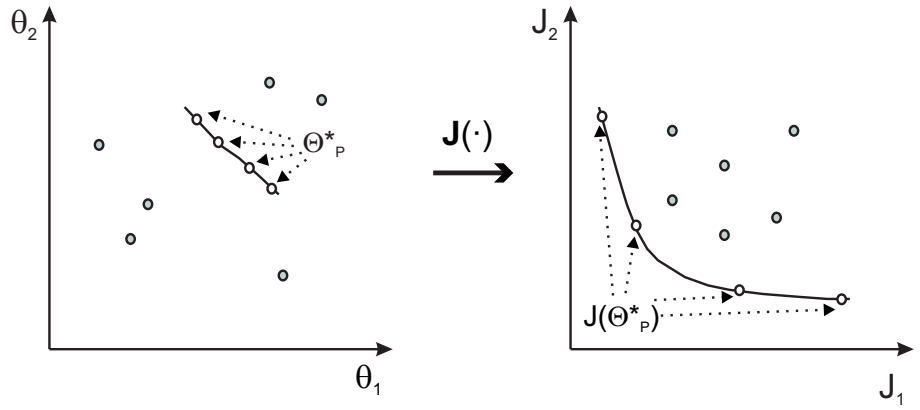

Figura 3.3: Muestra del conjunto de óptimos de Pareto Θ_P^* y la muestra distribuida del frente de Pareto que éstos producen $\mathbf{J}(\Theta_P^*)$ para un MOP de dos dimensiones.

A continuación, se procede a definir matemáticamente el mínimo global y local de un problema de optimización multiobjetivo:

Definición 3.6 (Mínimo global de un MOP): *Dado un vector de funciones* $\mathbf{J}(\cdot) : D \subseteq \mathcal{R}^L \rightarrow \Lambda \subseteq \mathcal{R}^s$, $D \neq \emptyset$, $s > 1$, *el conjunto* $\mathbf{J}(\Theta_P)$ *es llamado mínimo global si y sólo si*

$$\forall \theta \in D : \mathbf{J}(\Theta_P) \preceq \mathbf{J}(\theta),$$

donde $\Theta_P \in D$ *es el conjunto de vectores solución de mínimo global,* $\mathbf{J}(\cdot)$ *el vector de funciones de fitness y* D *el espacio de búsqueda de soluciones.*

$\triangle$

De la definición anterior se deduce que el frente de Pareto es el mínimo global de un MOP y las soluciones de dicho problema forman el conjunto de óptimos de Pareto.

Definición 3.7 (Mínimo local de un MOP): *Dado un vector de funciones* $\mathbf{J}(\cdot) : D \subseteq \mathcal{R}^L \rightarrow \Lambda \subseteq \mathcal{R}^s$, $D \neq \emptyset$, $s > 1$, *el conjunto* $\mathbf{J}(\Theta_L)$ *es llamado mínimo local si y sólo si*

$$\forall \theta \in \Theta_L, \nexists\, \theta' \in D : \theta' \preceq \theta \wedge ||\theta - \theta'|| < \epsilon \wedge ||\mathbf{J}(\theta) - \mathbf{J}(\theta')|| < \delta,$$

donde $\epsilon > 0$, $\delta > 0$ *y* $||\cdot||$ *representa una distancia métrica.*

$\triangle$

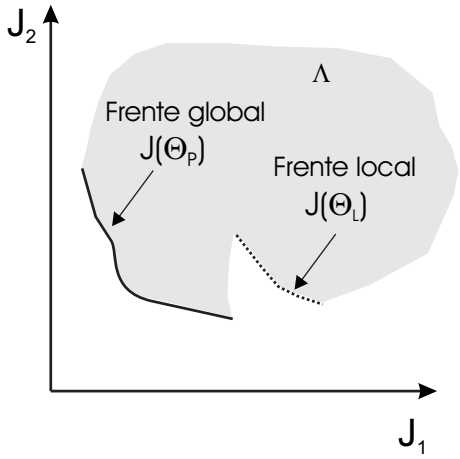

Figura 3.4: Espacio de funciones Λ. Frente de Pareto local (línea discontinua) $\mathbf{J}(\Theta_L)$ y global $\mathbf{J}(\Theta_P)$ (línea continua).

Por lo tanto, el mínimo local de un MOP constituirá un frente local de Pareto. La figura 3.4 muestra un ejemplo donde se puede distinguir entre el frente local y global de un MOP.

Para terminar con esta sección se define el vector ideal (ver figura 3.5).

Definición 3.8 (Vector Ideal): *Dado un vector de funciones* $\mathbf{J}(\theta) \in \Lambda \subseteq \mathcal{R}^s$, $\theta \in D \neq \emptyset$, $s > 1$, *se define el vector ideal* $\mathbf{J}^{ideal}$ *como:*

$$\mathbf{J}^{ideal} = [J_1(\theta^{*(1)}), J_2(\theta^{*(2)}), \ldots, J_s(\theta^{*(s)})],$$

donde $\theta^{*(i)}$, $i \in [1, \ldots, s]$ *denota el vector de soluciones que optimiza la función de fitness* J_i.

△

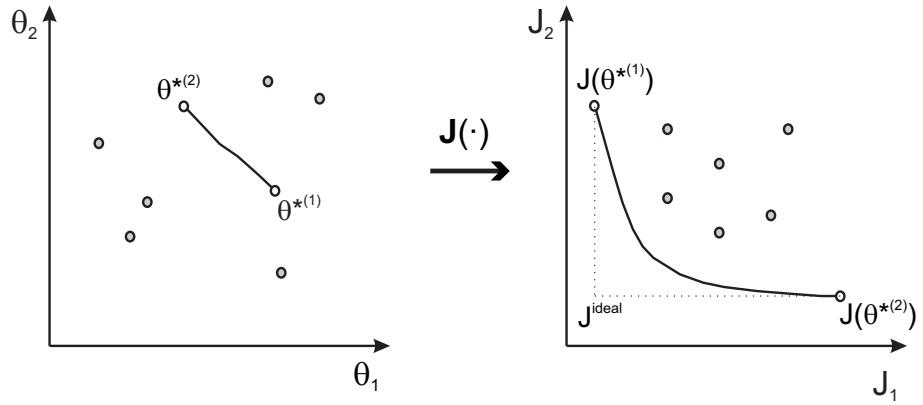

Figura 3.5: Vector ideal $\mathbf{J}^{ideal}$ para un MOP con dos objetivos J_1 y J_2 a minimizar. $\mathbf{J}(\theta^{*(1)}) = \{J_1(\theta^{*(1)}), J_2(\theta^{*(1)})\}$ y $\mathbf{J}(\theta^{*(2)}) = \{J_1(\theta^{*(2)}), J_2(\theta^{*(2)})\}$.

3.3. Métodos clásicos de resolución de MOP

La mayoría de los métodos clásicos de resolución de problemas de optimización multiobjetivo se fundamentan en la transformación de éste en otro problema de optimización global de una única función de *fitness* (SOP). Por lo general, el problema se reduce a la selección de una serie de parámetros (decisión) para convertir el MOP en un SOP y ejecutar la optimización (búsqueda), es decir, se tratan pues de técnicas *a priori*.

A continuación, se muestran algunos de los procedimientos clásicos más conocidos [124, 36, 80].

3.3.1. Combinación lineal de pesos

Este método transforma el MOP en SOP realizando una suma ponderada de los diferentes objetivos.

$$\min_{\theta \in D} \sum_{i=1}^{s} w_i J_i(\theta), \tag{3.1}$$

donde w_i son los pesos que ponderan la importancia de los diferentes objetivos $w_i \geq 0$ y $\sum_{i=1}^{s} w_i = 1$. La solución al problema 3.1 es un punto del frente de Pareto que, dependerá de los pesos escogidos y de las unidades en las que los diferentes objetivos estén expresados. Variando los pesos se irán consiguiendo diferentes puntos del frente con la desventaja que la solución puede resultar muy sensible a la modificación de los pesos. La solución será el punto de tangencia entre el frente de Pareto y la superficie equipotencial de valor mínimo para (3.1). Para el caso de dos dimensiones, se trata de una recta de pendiente

$$\frac{\pi}{2} + \arctan \frac{w_1}{w_2}.$$

La figura 3.6 muestra dos ejemplos de MOPs de dos dimensiones cuyos frentes de Pareto son convexos y no convexos respectivamente.

La parte (a) de la figura muestra la solución al problema propuesto con w_1 y w_2 para un frente de Pareto convexo, que corresponde al punto $\mathbf{J}^1$ del frente de Pareto. Modificando los pesos (por ejemplo aumentando w_1 hasta w_1', manteniendo el mismo w_2) sería posible dar con la solución $\mathbf{J}^2$. Sin embargo, en la parte (b) de la figura se puede observar que con los pesos w_1 y w_2 se alcanzaría la solución $\mathbf{J}^3$, mientras que la solución $\mathbf{J}^1$ (que es una solución del frente de Pareto) no podría darse para ningún valor positivo de w_1, w_2.

Por lo tanto, se concluye que para frentes de Pareto no convexos será imposible generar todos las soluciones óptimas de Pareto.

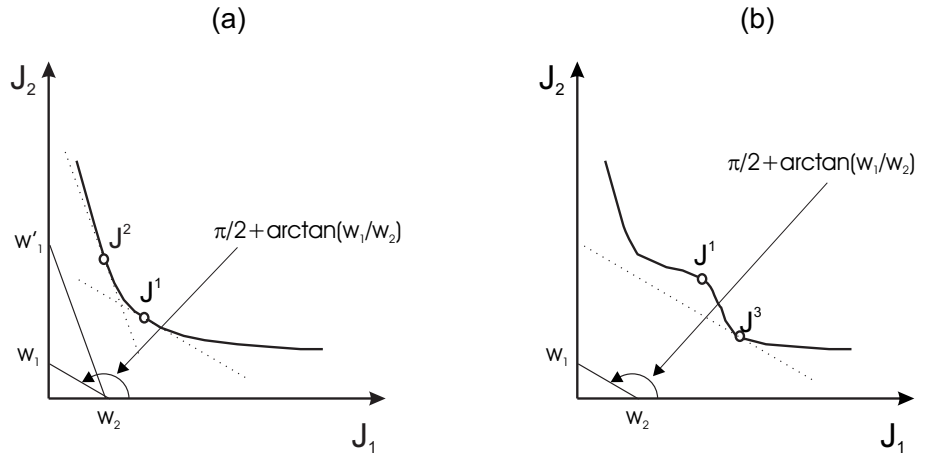

Figura 3.6: MOP mediante combinación lineal de pesos. (a) Frente de Pareto convexo. (b) Frente de Pareto no convexo.

3.3.2. Punto utópico

Este procedimiento plantea el MOP como la minimización de una función que mide la distancia de la solución a un determinado punto $J^u \in \mathcal{R}^s$

$$\min_{\theta \in D} ||\mathbf{J}^u - \mathbf{J}(\theta)||_p, \tag{3.2}$$

donde p representa la métrica, normalmente se utiliza $p = 1$ ó $p = 2$ aunque también se han utilizado otros valores. Un punto utópico típico es el punto ideal $\mathbf{J}^{ideal}$.

En ocasiones, se prefiere utilizar distancias relativas en lugar de absolutas

$$\min_{\theta \in D} \sqrt{\sum_{i=1}^{s} \left(\frac{J_i^u - J_i(\theta)}{J_i^u} \right)^p}^{1/p}. \tag{3.3}$$

Uno de los inconveniente de este método es que hay que definir el punto utópico y, si éste es el $\mathbf{J}^{ideal}$, para poder determinarlo habría que realizar s optimizaciones. Por otra parte, es posible dar con puntos situados en la parte cóncava del frente de Pareto.

3.3.3. Método de restricciones

Con este procedimiento se transforma un MOP con s objetivos:

$$\min_{\theta \in D} [J_1(\theta), J_2(\theta), \ldots, J_s(\theta)],$$

en un SOP con $s - 1$ restricciones:

$$\begin{aligned} \min_{\theta \in D} \quad & J_k(\theta) \\ s.a. \quad & J_i(\theta) \leq \epsilon_i \quad i \in [1, 2, \ldots, s],\ i \neq k. \end{aligned} \tag{3.4}$$

Al igual que con los pesos w_i (del método combinación lineal de pesos) las cotas ϵ_i pueden ser modificadas para dar con diferentes puntos del frente de Pareto tanto si es convexo como no.

Por ejemplo, para un MOP de dos dimensiones $s = 2$ y $k = 1$, el problema quedaría de la siguiente manera:

$$\begin{aligned} \min_{\theta \in D} \quad & J_1(\theta) \\ s.a. \quad & J_2(\theta) \leq \epsilon_2. \end{aligned}$$

La figura 3.7 muestra la solución que sería alcanzada para el caso de un frente de Pareto convexo y otro no convexo poniendo de manifiesto que podrían ser alcanzados puntos de la zona cóncava del frente.

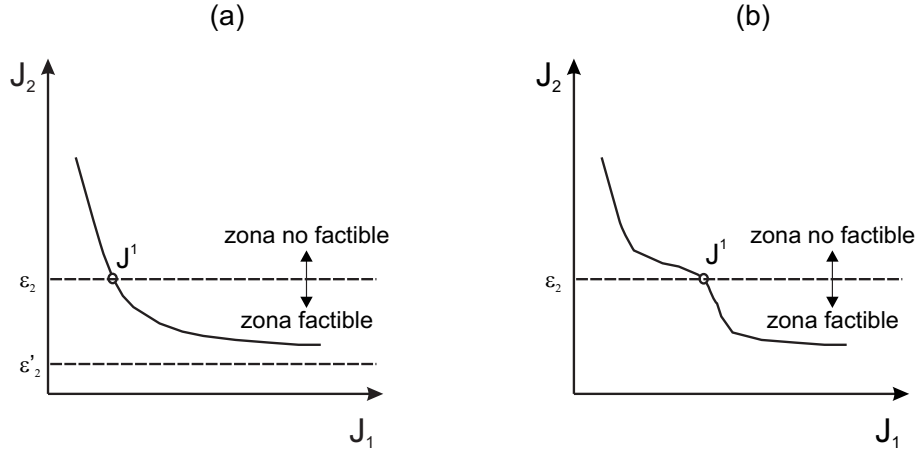

Figura 3.7: MOP mediante método de restricciones. (a) Frente de Pareto convexo. (b) Frente de Pareto no convexo.

Un inconveniente de la técnicas es que es necesario establecer unas cotas ϵ_i adecuadas, el problema podría no tener solución como se muestra en la figura 3.7 (a) cuando se plantea una cota ϵ_2'

$$\nexists\, \theta \in D : J_2(\theta) \leq \epsilon_2'.$$

Sin embargo, con este método es posible conseguir puntos en la zona cóncava del frente de Pareto.

3.3.4. Min-max

Este método transforma el MOP en un problema de optimización "min-max". Para ello se definen un vector, donde sus elementos miden de forma ponderada la adecuación a

cada objetivo. Se trata de minimizar con respecto al vector de soluciones el máximo valor del vector definido previamente. Es decir,

$$\min_{\theta \in D} \max[w_1 \cdot J_1(\theta), w_2 \cdot J_2(\theta), \ldots, w_s \cdot J_s(\theta)] \ , \tag{3.5}$$

donde $w_i \geq 0$, $i \in [1, \ldots, s]$ y $\sum_{i=1}^{s} w_i = 1$.

Aunque este método puede producir soluciones en la región cóncava del frente de Pareto (ver figura 3.8 (b)) no está garantizado que siempre produzca soluciones no dominadas.

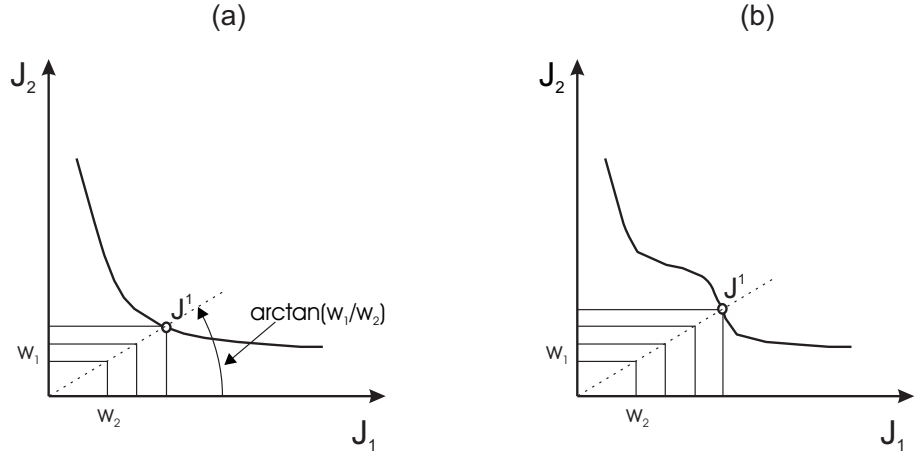

Figura 3.8: MOP mediante método "min-max" restricciones. (a) Frente de Pareto convexo. (b) Frente de Pareto no convexo.

La figura 3.8 muestra un ejemplo para un MOP de dos dimensiones (con frentes convexo y cóncavo). La solución al problema es el corte de la recta de pendiente

$$\arctan \frac{w_1}{w_2},$$

con el frente de Pareto.

Una variación de éste método, mide la distancia a un punto objetivo $\mathbf{J}^o$ introduciéndolo en (3.5) de la siguiente manera

$$\min_{\theta \in D} \max[w_1(J_1(\theta) - J_1^o), w_2(J_2(\theta) - J_2^o), \ldots, w_s(J_s(\theta) - J_1^o)] \ .$$

3.3.5. Lexicográfico

Esta técnica transforma el MOP en una secuencia de s optimizaciones O_i con restricciones. Para ello las componentes del vector de objetivos se ordenan según su importancia[4]

[4]En los métodos anteriores todas las funciones de *fitness* tenían la misma importancia aunque se utilizaban los pesos w_i para balancear la solución.

de manera que $J_1(\theta)$ es el objetivo de mayor importancia y por lo tanto $J_s(\theta)$ el menor. Así que la solución se obtiene minimizando las funciones $J_i(\theta)$ consecutivamente, empezando por la de mayor importancia.

El problema se formularía de la siguiente manera:

$$\begin{array}{ll} (O_1) & \text{mín}_{\theta \in D}\, J_1(\theta) = J_1^* \\ (O_2) & \text{mín}_{\theta \in D}\{J_2(\theta) \,|\, J_1(\theta) \leq J_1^*\} = J_2^* \\ & \vdots \\ (O_s) & \text{mín}_{\theta \in D}\{J_s(\theta) \,|\, J_1(\theta) \leq J_1^*, \ldots, J_{s-1}(\theta) \leq J_{s-1}^*\}. \end{array} \tag{3.6}$$

De una manera más compacta quedaría:

$$\begin{array}{c} \text{mín}_{\theta \in D}\, J_i(\theta) \\ s.a \quad J_j(\theta) \leq J_j^*, j \in [1, \ldots, i-1]. \\ i \in [1, \ldots, s] \end{array} \tag{3.7}$$

Evidentemente, este tipo de procedimientos tiene como inconveniente que es necesario determinar un orden de prioridad (importancia) de los diferentes objetivos, por lo tanto, sólo será aplicable en los casos en los que la prioridad de éstos sea evidente o esté determinada.

3.3.6. Discusión sobre los métodos clásicos

Lo que suele hacer que los métodos clásicos sean atractivos es que al transformar el MOP en un SOP las técnicas populares de optimización (ampliamente estudiadas y conocidas) pueden ser aplicadas. Evidentemente, los EAs podrían ser utilizados, de hecho lo han sido, para resolver los métodos clásicos planteados. Sin embargo, estos métodos tienen una serie de problemas o desventajas (independientemente de la herramienta de optimización utilizada):

- Las propias de los procedimientos *a priori*.
- Sólo generan un punto del frente de Pareto en cada ejecución del algoritmo siendo necesario repetir el procedimiento para obtener más puntos. Como las ejecuciones se realizan de forma independiente, la sinergia entre ellas no puede ser explotada y generalmente esto produce un mayor coste computacional.
- Asumen un conocimiento previo del problema (no siempre disponible) que se traduce en la selección de pesos o cotas o determinados puntos objetivo (utópico) a alcanzar. La sensibilidad a estos parámetros hace que el método sea ineficiente.

- Algunos algoritmos son sensibles a la forma (convexidad o concavidad) o continuidad del frente de Pareto.

3.4. Algoritmos evolutivos para MOP

La principal motivación del uso de los MOEAs es la posibilidad que éstos ofrecen de obtener, simultáneamente, un conjunto de soluciones óptimas de Pareto en una única ejecución del algoritmo. Además los MOEAs son menos sensibles a la forma del frente de Pareto, pudiendo obtener frentes no convexos y discontinuos. Los principales objetivos de los MOEAs (y en general de cualquier algoritmo de optimización multiobjetivo) son:

- Encontrar soluciones lo más cercanas posibles al frente de Pareto, a ser posible que estén en él. Lo que formalmente se conoce como una buena convergencia. En la mayoría de casos es difícil encontrar el conjunto de óptimos de Pareto Θ_P en un tiempo razonable, de ahí que el objetivo se reduzca a encontrar un conjunto $\Theta_P^* \subseteq \Theta_P$ de tamaño razonable, dando pie al siguiente objetivo.
- Que las soluciones estén lo mejor distribuidas a lo largo del frente para que lo caractericen adecuadamente. Lo que se conoce como una buena distribución.

Por lo tanto, un buen MOEA debería tratar ambas características de convergencia y distribución a lo largo del frente.

El uso de los EAs para resolver MOPs data de finales de la década de los sesenta. Desde entonces hasta ahora han sido innumerables las aportaciones realizadas en este campo. A continuación, se procede a describir los algoritmos más representativos en la historia de los MOEAs clasificados de la siguiente manera [35]:

- Métodos indirectos.
- Métodos directos.
  - No elitistas o MOEAs de primera generación.
  - Elitistas o MOEAs de segunda generación.

3.4.1. Métodos indirectos

Los algoritmos que se encuentran bajo esta categoría se caracterizan porque no utilizan el concepto de dominancia de Pareto (definición 3.9). Probablemente el primer EA aplicado a un MOP con la intención de determinar en una ejecución múltiples soluciones del frente de Pareto, sea VEGA [142]. Dada su relevancia a nivel histórico y su peculiaridad, a continuación se procede a detallar su funcionamiento.

VEGA

VEGA (Vector Evaluated Genetic Algorithm) modifica, respecto a un EA clásico, el proceso de selección, de manera que éste se repite s veces (tantas como objetivos tiene el MOP) tomándose cada vez un objetivo diferente. El proceso puede describirse de la siguiente manera:

- $i = 1$.
- Seleccionar N/s individuos de la población $P(t)$ (teniendo en cuenta la función de *fitness* J_i) y colocarlos en la subpoblación $P_i(t)$.
- $i = 1 + 1$. Si $i < s$ repetir paso anterior.
- $P'(t) = [P_1(t), P_2(t), \ldots, P_s(t)]$.
- Mezclar $P'(t)$ aleatoriamente.
- Aplicar los operadores de cruce y mutación sobre $P'(t)$ y generar $P(t+1)$.

VEGA presenta un comportamiento aceptable y su principal ventaja es su sencillez, pero su gran inconveniente es que tiende a concentrar la población en los extremos del frente de Pareto, lo que le imposibilita caracterizar correctamente todo el frente. Las subpoblaciones pueden tener el mismo número de individuos o distintos. El tamaño relativo de las subpoblaciones ejerce un efecto similar al de los pesos del algoritmo de combinación lineal de pesos.

Bajo esta categoría existen una serie de algoritmos, la mayor parte basados en las técnicas clásicas detalladas en la sección 3.3, que apenas han sido referenciados por los investigadores (a diferencia que VEGA). Por ejemplo en [88] se utiliza un algoritmo que utiliza la técnica de combinación lineal de pesos, donde éstos son modificados aleatoriamente, en [59] se presenta un algoritmo similar a VEGA que usa la optimización lexicográfica y selección por torneo y en [34] se propone un algoritmo basado en un punto utópico que se va modificando a partir de los individuos de la población.

3.4.2. Métodos directos

Para resolver el problema del algoritmo VEGA, en [63] se propuso por primera vez la aplicación del concepto de **no dominado** al proceso de *ranking* y selección.

Definición 3.9 (No dominado): *Un individuo θ^{nd} es no dominado en $P(t)$ si y sólo si:*

$$\nexists\, \theta \in P(t) : \theta \preceq \theta^{nd},$$

donde $P(t)$ representa el conjunto de individuos de la población P en la generación t.

△

Utilizando la definición anterior se crean una serie de niveles de dominancia. El primer nivel lo componen los individuos de la población que son no dominados. El segundo nivel

lo componen los individuos no dominados de la población, al que se le han eliminado los individuos del primer nivel y así sucesivamente hasta que a todos los individuos de la población se les ha asignado un determinado nivel de dominancia o *ranking* (ver figura 3.9).

De esta manera, el proceso de selección tiene en cuenta el nivel de dominancia del individuo que, en definitiva, es un indicador de su distancia respecto del frente de Pareto.

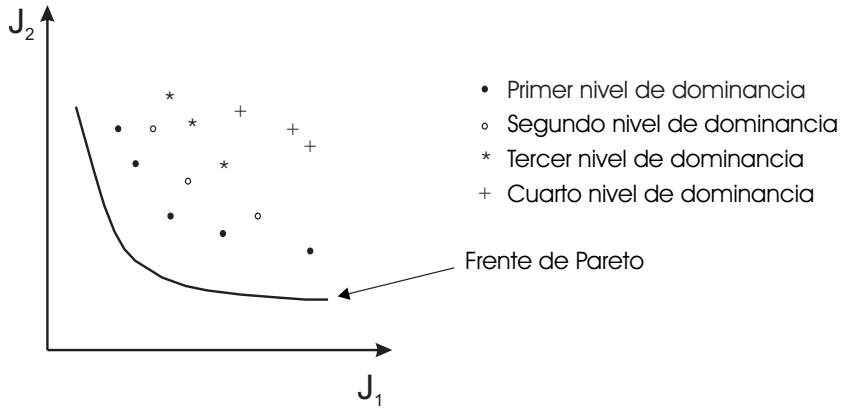

Figura 3.9: Ejemplo de asignación del nivel de dominancia en un MOP de dos dimensiones.

Esto marcaría un antes y un después en la evolución de los MOEAs. En [63] también se planteó la necesidad de incorporar mecanismos que evitasen la convergencia prematura.

Los MOEAs que se detallan bajo la categoría de métodos indirectos se caracterizan por utilizar, de un modo u otro, el concepto de no dominado o dominancia de Pareto. Estos métodos pueden ser agrupados en torno a dos generaciones de MOEAs:

- Una primera generación donde se intenta evitar el problema de convergencia introduciendo mecanismo de *fitness* compartido (ver sección 2.11.1.2) o generación de nichos.
- Una segunda generación caracterizada por la noción de elitismo.

3.4.2.1. No elitistas

A continuación, se procede a mostrar las características de los MOEAs más importantes bajo las categorías de no elitistas (primera generación).

MOGA

MOGA (*Multiobjective Genetic Algorithm*) [58] está fundamentado en el concepto de no dominado, ya comentado y, asigna un valor de *ranking* a cada individuo que es el número de individuos de la población que le dominan más uno.

De esta manera, todos los individuos no dominados tienen un valor de *ranking* de uno, mientras que los dominados son penalizados (aumentado su *ranking*) en función del número de individuos que les dominan (ver figura 3.10).

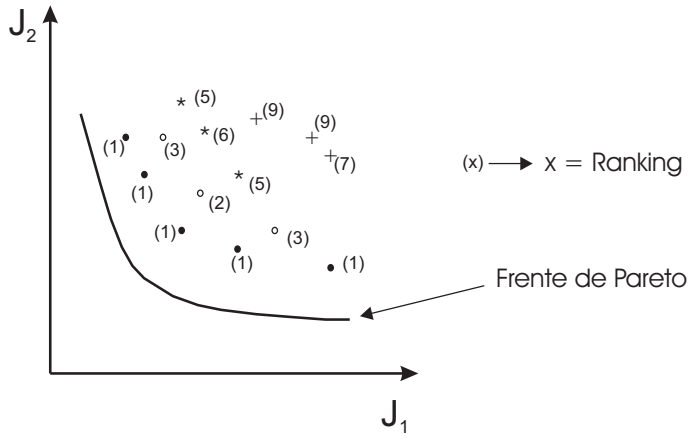

Figura 3.10: Ejemplo de asignación del valor de *ranking* para el mismo MOP de la figura 3.9.

Después de ser ordenado de acuerdo al valor del *ranking*, a los individuos se les asigna un valor de *fitness* interpolando linealmente desde 1 hasta el de valor máximo de *ranking* (en el caso particular de la figura 3.10 sería 9) como se sugiere en [63].

Adicionalmente, MOGA incorpora un mecanismo de *fitness* compartido con el objetivo de reducir la convergencia prematura. La distancia entre individuos se mide en el espacio de objetivos ya que, lo que se pretende es obtener un frente de Pareto lo más distribuido posible. El medir distancia entre individuos en el espacio de objetivos tiene como principal inconveniente que el algoritmo intente impedir que dos individuos diferentes θ' y θ'' para los que $\mathbf{J}(\theta') = \mathbf{J}(\theta'')$ coexistan en la población. Como principales ventajas es su sencillez y es bastante eficiente. Por otra parte, al incorporar la técnica de nichos el resultado es muy dependiente de parámetro *radio del nicho*.

NPGA

NPGA (*Niched Pareto Genetic Algorithm*) [86] combina el concepto de nivel de dominancia y la técnica de nichos con el procedimiento de selección por torneo (ver 2.5.2).

La selección por torneo entre dos individuo θ^1 y θ^2 se resuelve de la siguiente manera:

- Se seleccionan n individuos de la población aleatoriamente.
- Se compara θ^1 y θ^2 con los n individuos.
- Si θ^1 no es dominado por ninguno de los n individuos y θ^2 sí, se selecciona el individuo θ^1.
- Si θ^2 no es dominado por ninguno de los n individuos y θ^1 sí, se selecciona el individuo θ^2.

- Si no se da ninguno de los dos casos anteriores, se escoge el individuo que presente menor densidad de nicho.

Los parámetros n y el *radio del nicho* son parámetros cruciales en el funcionamiento del algoritmo. En cuanto al espacio para medir las distancias entre individuos, se han utilizado ambos, el espacio de soluciones y el de objetivos, siendo este último el más efectivo.

Aunque el algoritmo presente un buen comportamiento, el uso de la técnica de nichos empeora los reducidos costes computacionales producidos por el proceso de selección basado en torneo.

NSGA

NSGA (*Nondominated Sorting Genetic Algorithm*) [148, 40] utiliza también el concepto de nivel de dominancia definido anteriormente.

Para una población de N individuos, se les asigna a los elementos del primer nivel de dominancia, un valor de *fitness* provisional igual a N. Este valor es modificado utilizando la técnica de *fitness* compartido (ver sección 2.11.1.2). A continuación, al segundo nivel de dominancia se le asigna un valor de *fitness* provisional algo inferior al peor valor de *fitness* de los individuos del primer nivel de dominancia y así sucesivamente. El proceso de selección que utiliza es el de ruleta múltiple (ver sección 2.5.1). Algunos autores [30] clasifican NSGA como una versión de MOGA más sensible a parámetros como el *radio del nicho*.

3.4.2.2. Elitistas

El concepto de algoritmo elitista [106] determina que los individuos mejores (los no dominados) no pueden ser eliminados de la población en favor de individuos peores (dominados). Este concepto puede definirse formalmente como sigue:

Definición 3.10 (Elitista): *Sea $P(t)$ la población de individuos de un EA en la generación $t \in \mathcal{N}$. Sea $prob(\theta)^t$ la probabilidad de que un individuo $\theta \in P(t)$ sea seleccionado como un operando de un operador de variación en la generación t. El EA será elitista, si y sólo si para cualquier relación de preferencia ($\preceq$) dada por un problema de decisión, se mantiene la siguiente condición:*

$$\forall t \in \mathcal{N} : \exists\, \theta \in \mathcal{P}^*(t) \mid prob(\theta)^{t+1} > 0,$$

donde $\mathcal{P}^*(t)$ representa todos los individuos no dominados de

$$\bigcup_{\tau \leq t} \mathcal{P}(\tau)\ ,$$

es decir, todos los individuos producidos durante el proceso de optimización.

$\triangle$

Normalmente aunque un MOEA no utilice el concepto de elitista definido en 3.10, suele ser bastante habitual que las mejores soluciones (las no dominadas $\mathcal{P}^*(t)$) se almacenen en un archivo $A(t)$ de manera que, éste representará la mejor aproximación al frente de Pareto $\mathbf{J}(\Theta_P)$. Dado que el archivo se utiliza sólo como un almacén de individuos el comportamiento del EA no se ve alterado. Por contra, si el archivo actúa de forma activa en el proceso de optimización, facilitando la explotación de la información contenida en él, el MOEA se considerará elitista.

El uso de un archivo implica una serie de cuestiones:

- Interacción entre el archivo y la población.
- Procedimiento a aplicar cuando el archivo se llena.
- Criterios para introducir individuos en el archivo, además del criterio de dominancia de Pareto.

A continuación, se procede a detallar los MOEAs elitistas más representativos.

SPEA

SPEA (*Strength Pareto Evolutionary Algorithm*) [168] utiliza también el concepto de dominancia y el de elitismo mediante la incorporación de un archivo $A(t)$ que actúa de forma activa. La figura 3.11 muestra un esquema del funcionamiento del algoritmo que se describe a continuación:

- Los elementos no dominados de $P(t) \bigcup A(t)$ son almacenados en el archivo $A(t+1)$[5]. Si el número de individuos de $A(t+1)$ excede el límite establecido para el archivo, se eliminan los individuos sobrantes utilizando una técnica de clustering que preserva las características del frente creado por los individuos de $A(t+1)$.
- A cada individuo del archivo se le asigna un valor de *fitness* relacionado con el número de individuos de la población $P(t)$ que domina dividido por el número de individuos más uno.
- A cada individuo de $P(t)$ se le asigna un valor de *fitness* que se obtiene de sumar el valor de *fitness* de todos los individuos del archivo que lo dominan más uno.
- A continuación, sobre $P(t) \bigcup A(t+1)$ se produce el proceso de selección de los individuos que constituirán $P(t+1)$ (una vez sean cruzados y mutados) utilizando un método de selección por torneo con dos individuos.

El comportamiento del SPEA es bastante bueno aunque el coste computacional resulta muy elevado comparado con el resto de algoritmos.

[5]Inicialmente $P(0)$ se crea de forma aleatoria y el archivo $A(0) := \emptyset$.

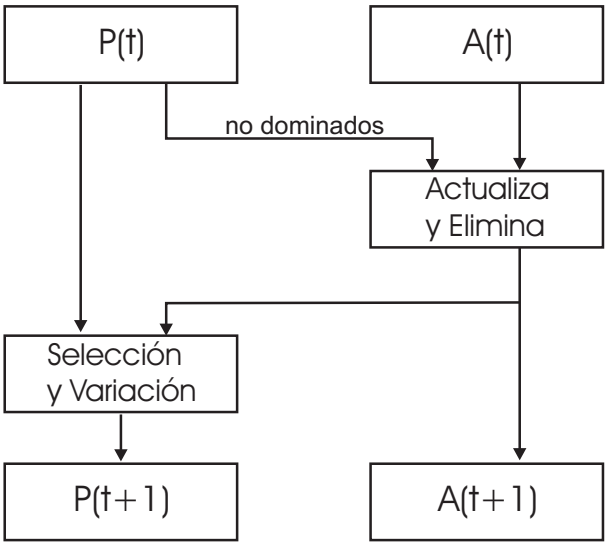

Figura 3.11: Esquema de la secuencia de SPEA.

SPEA 2

En [170] se presenta una nueva versión de SPEA, llamada SPEA 2 que incorpora la siguientes tres diferencias:

- Para evitar que individuos en $P(t)$ dominados por el mismo número de individuos del archivo, tengan el mismo valor de *fitness*, se tiene en cuenta también el número de individuos que éstos dominan.
- El procedimiento de eliminación de individuos del archivo, basado en una técnica de *clustering*, se ve modificado por otro que va eliminando aquellos individuos que estén más próximos a otros.
- Asegura que los individuos que se encuentren en los extremos del frente no son eliminados, con el objetivo de mejorar la caracterización del mismo.

Al igual que ocurría con SPEA, SPEA 2 presenta también costes computaciones elevados aunque mejora en prestaciones a su antecesor [170].

NSGA II

Una versión más sofisticada del NSGA, el NSGA II [43] introduce las siguientes mejoras:

- Reduce los costes computacionales en el proceso de determinación de los niveles de dominancia.
- Sustituye la técnica de *fitness* compartido por el mecanismo de apareamiento restringido (ver sección 2.11.1.3).
- Introduce el enfoque elitista haciendo competir a los padres con sus hijos (EA solapable) mediante selección por torneo, pero no introduce un archivo.

PAES

PAES (*Pareto Archived Evolution Strategy*) [98] es un (1+1)-ES[6] junto con un archivo que va almacenando los individuos no dominados que se van encontrando.

El tratamiento del archivo es lo que caracteriza a PAES y lo diferencia de los algoritmos anteriores. Para mantener diversidad en el algoritmo, PAES divide el espacio de objetivos mediante un *grid* (con un determinado número de divisiones en cada dimensión). El algoritmo mantiene un contador de los individuos que hay en cada *grid*. De esta manera, cuando el límite del archivo ($refpop$) es superado se elimina una de las soluciones que se encuentre en el *grid* que más individuos contenga.

Por otra parte, a la hora de seleccionar un individuo del archivo, para que actúe como padre en el proceso de reproducción, se escoge el individuo que permanezca en un *grid* cuyo contador sea menor. De esta manera, se fomenta la exploración de zonas menos densas para intentar generar diversidad en el archivo, que constituirá una aproximación del frente de Pareto con un número de individuos inferior a $refpop$.

La ventaja de PAES frente a los algoritmos anteriores es que los costes computaciones son bastante menores que los producidos por los algoritmos que utilizan *ranking* basado en los niveles de dominancia. Sus principal inconveniente es que no funciona adecuadamente en frentes de Pareto discontinuos, motivado en parte por la característica de optimizador local que presenta.

Existen versiones de PAES con (μ+1)-ES[7] y (μ+λ)-ES[8] aunque, en general, esto no aumenta las prestaciones del algoritmo y sí los costes computacionales.

PESA

PESA (*Pareto Envelope-based Selection Algorithm*) [38] incorpora ideas de SPEA y PAES. Al igual que éstos, utiliza una archivo que almacena la aproximación al frente de Pareto y una población (de menor tamaño que el archivo) que contiene nuevos individuos que son candidatos a ser incluidos en el archivo.

Igual que PAES, PESA utiliza un *grid* para dividir el espacio de los objetivos y mantiene un registro de las soluciones que se mantienen en cada *grid* para generar diversidad en la aproximación al frente de Pareto. Las principales diferencias entre PAES y PESA son:

- PESA utiliza operadores de cruce y mutación mientras que PAES sólo mutación.
- En el proceso de selección, PAES actúa sobre un individuo y su descendiente, mientras que PESA selecciona los individuos del archivo utilizando por una parte una componente aleatoria y por otra el contador del *grid*.

Al igual que PAES, presenta problemas en la caracterización de los bordes del frente de Pareto.

[6] Un único individuo genera otro, mediante mutación y ambos compiten entre sí por sobrevivir.
[7] A partir de μ se genera un hijo.
[8] A partir de μ padres se generan λ hijos.

PESA II Existe una revisión de PESA, llamada PESA II [37] muy similar a su predecesor con la diferencia que la selección se realiza de la siguiente forma:

- Se selecciona aleatoriamente un *grid*.
- A continuación, se selecciona aleatoriamente un individuo perteneciente a dicho *grid*.

3.4.3. Comparación

En esta sección se presentan simplemente y para que sirva como referencia, algunos de los trabajos donde se comparan los diferentes MOEAs que se han presentado en las secciones anteriores.

En [168] y [169] se muestra una comparación entre SPEA, NPGA, NSGA, MOGA y VEGA, utilizando seis funciones de test creadas con la estructura propuesta en [46] y se clasifica los MOEAs en el siguiente orden SPEA, NSGA, VEGA, NPGA, MOGA, de mayor a menor prestaciones.

En [97] se compara PAES con SPEA utilizando las mismas seis funciones de test y se concluye que PAES supera a SPEA en 4 de ellas, mientras que en las otras dos SPEA supera a PAES.

En [43] se compara NSGA II con PAES utilizando 5 funciones test mostrando que NSGA II supera en prestaciones a PAES.

En [38] se compara PESA con PAES y SPEA concluyendo que PESA supera a PAES y SPEA en 4 de las 6 funciones de test utilizadas, PESA y SPEA presentan las mismas prestaciones en una de ellas y SPEA supera a PESA y PAES en la otra.

En [170] se compara SPEA 2 con SPEA, PESA y NSGA II concluyendo:

- SPEA 2 mejora a su predecesor SPEA.
- PESA presenta una convergencia más rápida pero tiene dificultades para mantener los extremos del frente.
- SPEA 2 y NSGA II muestran un comportamiento muy similar aunque SPEA 2 lo supera cuando el espacio de objetivos tiene una dimensión elevada.

En general, es difícil determinar qué algoritmo presenta mejores prestaciones, ya que las comparaciones no siempre son aclaratorias y hay muchos aspectos que resultan clave como las funciones de test utilizadas, el ajuste de los parámetros de los diferentes MOEAs en estudio o el coste computacional, que generalmente quedan encubiertos.

Por lo general, parece evidente que los MOEAs de segunda generación (elitistas) superan a los de la primera generación (no elitistas).

En estos momentos los investigadores en esta materia están planteándose cuáles serán los algoritmos que constituirán la tercera generación. En [33] se apunta a una generación

de algoritmos eficientes y con estructuras de datos en el espacio que permitan aumentar la eficacia del almacenaje en el archivo.

3.5. Métricas

En esta sección se van a presentar los aspectos relacionados con las medidas que permitirán validar y cuantificar las prestaciones de los MOEAs en estudio. El análisis de las métricas resulta especialmente crucial, dado que en un MOP la solución no será única, sino una aproximación al conjunto de óptimos de Pareto. Por otra parte, la propia naturaleza estocástica de los EAs hace necesaria la ejecución del algoritmo varias veces para poder determinar sus prestaciones reduciendo así el efecto de aleatoriedad, de ahí que sea necesario utilizar herramientas de análisis estadístico.

Normalmente, las métricas producidas por los investigadores han ido encaminadas a determinar los siguientes aspectos [36]:

- Medir la distancia entre la aproximación al frente de Pareto $\mathbf{J}(\Theta_P^*)$, obtenido por el algoritmo y el propio frente de Pareto $\mathbf{J}(\Theta_P)$, asumiendo que este último es conocido[9].
- Cuantificar la calidad (uniformidad, extensión, número de puntos, etc.) de la distribución de $\mathbf{J}(\Theta_P^*)$ sobre $\mathbf{J}(\Theta_P)$.

La determinación de las métricas no es una tarea trivial y es prácticamente imposible determinar una única métrica que permita caracterizar todos los criterios anteriores significativamente. A continuación, se muestran algunas de las métricas más utilizadas [36, 169]:

Métricas de convergencia

- **Ratio de Error (RE):** Mide el porcentaje de individuos de $\mathbf{J}(\Theta_P^*)$ que son parte de $\mathbf{J}(\Theta_P)$. Matemáticamente puede ser representado de la siguiente manera:

$$RE := \frac{\sum_{i=1}^{n} e(\theta_i)}{n}, \tag{3.8}$$

donde n es el número de individuos de $\mathbf{J}(\Theta_P^*)$ y

$$e(\theta_i) = \begin{cases} 0, & \text{si } \mathbf{J}(\theta_i) \in \mathbf{J}(\Theta_P) \\ 1, & \text{otro caso} \end{cases}.$$

Por lo tanto, $RE = 1$ indica una mala convergencia, mientras que $RE = 0$ indica que $\mathbf{J}(\Theta_P^*) \subseteq \mathbf{J}(\Theta_P^*)$.

[9] Es posible obtener una buena aproximación de Θ_P, que sirva como referencia, aplicando una búsqueda exhaustiva con un *grid* muy fino, a costa de emplear unos recursos computacionales elevados.

- **Distancia Generacional (DG):** Mide la distancia, en términos medios, entre $\mathbf{J}(\Theta_P^*)$ y $\mathbf{J}(\Theta_P)$ de la siguiente manera[10]:

$$DG := \frac{\sum_{i=1}^{n} d(\theta_i)}{n}, \tag{3.9}$$

donde n es el número de individuos de $\mathbf{J}(\Theta_P^*)$ y $d(\theta_i)$ es la distancia Euclídea (en el espacio de objetivos) entre cada punto $\mathbf{J}(\theta_i)$ de $\mathbf{J}(\Theta_P^*)$ y el punto de $\mathbf{J}(\Theta_P)$ más cercano.

$$d(\theta_i) = \min_{\theta \in \Theta_P} ||\mathbf{J}(\theta_i) - \mathbf{J}(\theta)||_2. \tag{3.10}$$

La misma métrica puede ser utilizada para determinar el proceso de convergencia si $d(\theta_i)$ mide la distancia entre $\mathbf{J}(\Theta_P^*(t))$ y el individuo más cercano de $\mathbf{J}(\Theta_P^*(t-1))$. A partir de la distancia generacional se define otra métrica denominada **Medida de Progreso (MP)** que, como su propio nombre indica, se utiliza para evaluar la convergencia en la iteración t comparándola con la de la primera iteración.

$$MP := \sqrt{\frac{DG(1)}{DG(t)}}, \tag{3.11}$$

donde $DG(1)$ es la distancia generacional en la generación 1 y $DG(t)$ en la generación t.

- **Máximo Error sobre el Frente de Pareto (ME):** Esta métrica indica la máxima de las distancias mínimas de cada individuo en $\mathbf{J}(\Theta_P^*)$ al individuo más cercano del frente de Pareto $\mathbf{J}(\Theta_P)$.

$$ME := \max_{\theta \in \Theta_P} \min_{\theta \in \Theta_P^*} ||\mathbf{J}(\theta^*) - \mathbf{J}(\theta)||_2. \tag{3.12}$$

Por lo tanto, $ME = 0$ indica que $\mathbf{J}(\Theta_P^*) \subseteq \mathbf{J}(\Theta_P)$ y cualquier otro valor indica que al menos un individuo de $\mathbf{J}(\Theta_P^*) \nsubseteq \mathbf{J}(\Theta_P)$. A mayor valor de ME mayor es la distancia mínima al frente de Pareto.

- **Individuos no dominados añadidos (INA):** Mide la cantidad de individuos no dominados que son encontrados en cada generación.

$$INA := |\mathbf{J}(\Theta_P^*(t))| - |\mathbf{J}(\Theta_P^*(t-1))|, \tag{3.13}$$

donde $|\cdot|$ indica el número de individuos del conjunto.

[10] En [169] DG es denominada como $\mathcal{M}_1$.

Métricas de distribución, extensión, etc.

- **Espaciado (SP):** Esta métrica mide la varianza de la distancia entre los individuos vecinos de $\mathbf{J}(\Theta_P^*)$ de la siguiente manera:

$$SP := \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d(\theta_i))}, \tag{3.14}$$

  donde d_i se obtiene como:

$$d(\theta_i) = \min_{\theta_j \in \Theta_P^*} ||\mathbf{J}(\theta_i) - \mathbf{J}(\theta_j)||_1. \tag{3.15}$$

  $\overline{d}$ es la media de todos los $d(\theta_i)$, n es el número de individuos en $\mathbf{J}(\Theta_P^*)$. $S = 0$ indica que todos los individuos de $\mathbf{J}(\Theta_P^*)$ permanecen equidistantes. A la hora de aplicar esta métrica, hay que tener en cuenta que puede que el frente de Pareto $\mathbf{J}(\Theta_P)$ no tenga una distribución uniforme, por ejemplo, cuando éste es discontinuo. En este caso, la métrica debería aplicarse a las diferentes porciones del frente de forma separada.

- $\mathcal{M}_2$**:** Mide la distribución en combinación con el número de individuos no dominados encontrados:

$$\mathcal{M}_2 := \frac{1}{n-1}\sum_{i=1}^{n} q(\theta_i), \tag{3.16}$$

  donde n es el número de individuos de $\mathbf{J}(\Theta_P^*)$ y

$$q(\theta_i) = \begin{cases} 0, & \text{si } \exists\, \theta \in \mathbf{J}(\Theta_P^*) : ||\mathbf{J}(\theta_i) - \mathbf{J}(\theta)|| > \sigma \\ 1, & \text{Otro caso} \end{cases},$$

  siendo $\sigma > 0$ un parámetro de vecindad (distancia entre nichos) y $||\cdot||$ una distancia métrica. Evidentemente, cuanto más grande sea $\mathcal{M}_2$ mejor será la distribución dado un determinado σ.

- **Individuos no dominados (IND):** Con esta métrica simplemente se contabilizan el número de individuos no dominados, es decir los individuos en Θ_P^*.

$$IND := |\Theta_P^*|. \tag{3.17}$$

- **Hiper-área (H):** Mide el área bajo la curva definida por el frente de Pareto.

$$H := \{\bigcup_i a(\theta_i) \mid \theta_i \in \Theta_P^*\}, \tag{3.18}$$

  donde $a(\theta_i)$ es el área contenida en el *box* definido por la coordenada $\mathbf{J}(\theta_i)$ y el origen. A partir de H se puede definir el ratio de Hiper-área (HR) como:

$$HR := \frac{H(\Theta_P^*)}{H(\Theta_P)} \tag{3.19}$$

es decir, el ratio entre el Hiper-área producida por la aproximación al frente de Pareto y el Hiper-área producido por el propio frente de Pareto. De manera que, $HR = 1$ es equivalente a $\mathbf{J}(\Theta^*_P) = \mathbf{J}(\Theta_P)$ y HR irá creciendo a medida que $\mathbf{J}(\Theta^*_P)$ se aleje del frente de Pareto $\mathbf{J}(\Theta_P)$.

- $\mathcal{M}_3$**:** Mide la extensión del frente $\mathbf{J}(\Theta^*_P)$ en cada dimensión, utilizando la siguiente expresión:

$$\mathcal{M}_3 := \max_{\theta_i,\theta_j \in \Theta^*_P} ||\mathbf{J}(\theta_i) - \mathbf{J}(\theta_j)||_2. \tag{3.20}$$

  Para el caso de un MOP de dos objetivos, es la distancia entre los dos extremos de la aproximación al frente de Pareto.

Métrica de comparación

- **Alcance comparativo ($\mathcal{C}$):** Dadas dos aproximaciones al frente de Pareto $\mathbf{J}(\Theta^*_{P1})$ y $\mathbf{J}(\Theta^*_{P2})$ generados por los dos MOEAs que se desean comparar, $\mathcal{C}$ mapea éstas en el intervalo $[0, 1]$ utilizando la siguiente expresión:

$$\mathcal{C}(\Theta^*_{P1}, \Theta^*_{P2}) := \frac{|\theta_i \in \Theta^*_{P2}; \exists\, \theta_j \in \Theta^*_{P1} : \theta_j \preceq \theta_i|}{|\Theta^*_{P2}|}. \tag{3.21}$$

  Por lo tanto un valor $\mathcal{C}(\Theta^*_{P1}, \Theta^*_{P2}) = 1$ implica que todos los individuos de Θ^*_{P2} son dominados por algún individuo de Θ^*_{P1} y $\mathcal{C}(\Theta^*_{P1}, \Theta^*_{P2}) = 0$ implica justamente lo contrario. Dado que $\mathcal{C}(\Theta^*_{P1}, \Theta^*_{P2})$ no es necesariamente igual a $1 - \mathcal{C}(\Theta^*_{P1}, \Theta^*_{P2})$, también debería ser calculado $\mathcal{C}(\Theta^*_{P2}, \Theta^*_{P1})$. La ventaja de esta métrica es que es bastante fácil de calcular y muestra el resultado de un algoritmo domina al resultado del otro.

Dado la componente estocástica de los EAs, las métricas anteriores suelen calcularse varias veces sobre ejecuciones diferentes del algoritmo estableciendo resultados estadísticos, los más comunes: máximos y mínimos, medias, mediana, varianza, etc., que suelen representarse gráficamente.

3.6. Conclusiones

A lo largo del capítulo se han tratado los conceptos básicos y la terminología asociada a los problemas de optimización multiobjetivo (MOPs). Se han presentado los métodos clásicos de resolución de MOPs más importantes así como los MOEAs más utilizados en la resolución de MOPs clasificándolos como de primera (no elitistas) y segunda generación (elitistas). Los MOEAs que se han presentado, gracias a la propia naturaleza poblacional de los mismos son capaces de generar, al mismo tiempo, varios elementos del conjunto de óptimos de Pareto. Esto posibilita la selección de la solución *a posteriori* una vez estudiadas las posibilidades que ofrecen las soluciones de Pareto. También se han presentado una serie de métricas que permitirán analizar el funcionamiento de los MOEAs y compararlos.

Sin duda, la potencia y flexibilidad de los MOEAs despierta el interés de los investigadores, haciendo que el campo de la computación multiobjetivo mediante EAs sea un campo en boga dentro del área de la inteligencia artificial, sirva como ejemplo los siguientes trabajos que aún se encuentran en prensa [47, 5, 114].

Capítulo 4

Algoritmo Evolutivo $\not\epsilon^{\nearrow}$-MOGA

4.1. Introducción

El capítulo anterior presentó los conceptos básicos relacionados con la optimización multiobjetivo y los MOEAs más importantes de la primera y la segunda generación, elitistas y no elitistas respectivamente. Este capítulo tiene como principal objetivo presentar la implementación de un MOEA denominado ϵ-MOGA inspirado en las directrices dadas en [105, 104]. El objetivo de ϵ-MOGA es garantizar que la solución Θ_P^* que genere, converja hacia el conjunto de óptimos de Pareto Θ_P ($\Theta_P^* \subseteq \Theta_P$) utilizando memoria limitada[1] (lo que comúnmente se conoce como convergencia del algoritmo) y al mismo tiempo intentar conseguir una buena distribución de la solución $\mathbf{J}(\Theta_P^*)$ a lo largo de $\mathbf{J}(\Theta_P)$.

Para que un algoritmo evolutivo, en general, pueda asegurar convergencia es necesario que cumpla las siguientes condiciones [141]:

1. El operador genético de mutación debe poder generar cualquier solución $\theta' \in D$ a partir de cualquier solución $\theta \in D$ con probabilidad mayor que cero.

2. Debe asegurar que una solución óptima no se pierda, es decir, no se produzca deterioro[2].

La primera de las condiciones es muy sencilla de cumplir ya que, los operadores genéticos de mutación presentados en la sección 2.6.2 la cumplen. La segunda de las condiciones sería sencilla de cumplir en el caso de optimización de una única función de *fitness*, simplemente utilizando un EA elitista, pues sólo hay que almacenar una solución (la mejor) y ésta sólo se reemplaza por otra que la mejore. Para el caso MO no es suficiente con que el MOEA sea elitista, ya que soluciones no dominadas podrían ser descartadas debido a que el archivo $A(t)$ podría superar sus límites. Por lo tanto, la garantía de la convergencia en problemas MOEA pasa por utilizar un procedimiento de actualización del archivo $A(t)$ adecuado.

Existen estrategias que consiguen evitar el deterioro y garantizar convergencia (ver [105] y sus referencias) sin embargo, no utilizan mecanismos que intenten conseguir una buena distribución o cuando el archivo $A(t)$ se llena de soluciones no dominadas no permite introducir nuevas soluciones en él.

Por otra parte, los algoritmos de la segunda generación presentados en el capítulo anterior (PAES, SPEA 2, NSGA II y PESA II) aunque sí que fomentan la generación de frentes de Pareto distribuidos no consiguen asegurar convergencia ya que la gestión de las soluciones no dominadas, mantenidas por los diferentes algoritmos, hacen referencia sólo a criterios que miden la densidad de las soluciones existentes.

Sin embargo, en [105] se plantea una estrategia que garantiza la determinación de una aproximación Θ_{P_ϵ} que garantiza convergencia fomentado distribución a lo largo del

[1] Dado que Θ_P puede contener un conjunto infinito de soluciones.

[2] El deterioro ocurre cuando en un instante t una solución $\theta' \in A(t)$ es dominada por una solución θ^* que perteneció a $A(t)$ en un instante anterior a t. $A(t)$ se encarga de almacenar la aproximación Θ_P^* a Θ_P

frente de Pareto. Una implementación de esta estrategia se presenta en [45] donde se ponen de manifiesto, a través de una serie de funciones de test, las propiedades de no deterioro del algoritmo, la diversidad producida y la rapidez del mismo comparándolo con los algoritmos PESA, SPEA, NSGA II y una versión de NSGA II con *clustering*.

El resto del capítulo se distribuye de la siguiente manera: la sección 4.2 describe la estrategia planteada en [105] para dar con una aproximación del frente de Pareto de tamaño finito. En la sección 4.3 se presenta el algoritmo ϵ-MOGA desarrollado para obtener con Θ_{P_ϵ} destacando las principales diferencias con el algoritmo presentado en [45]. La sección 4.4 evalúa el algoritmo ϵ-MOGA utilizando un conjunto de funciones de test [36]. La siguiente sección propone un algoritmo nuevo $\not\epsilon$MOGA como mejora del algoritmo ϵ-MOGA y lo evalúa utilizando las mismas funciones de test. En la sección 4.6 se aplica $\not\epsilon$MOGA sobre un ejemplo de ingeniería donde se plantea un problema de optimización multiobjetivo aplicado a una estructura mecánica con tres barras y por último la sección 4.7 muestra las conclusiones más importantes de este capítulo.

4.2. Conceptos relacionados con ϵ-dominancia

Antes de describir el algoritmo $\epsilon - MOGA$ se procede a definir, a lo largo de esta sección, la terminología asociada al concepto ϵ-dominante que servirá para dar con una determinada aproximación del frente de Pareto[3].

Definición 4.1 (ϵ-dominante): *Dado dos vectores θ^1, θ^2 en el espacio de soluciones, cuyas imágenes en el espacio de las funciones de fitness son $\mathbf{J}(\theta^1)$ y $\mathbf{J}(\theta^2)$ respectivamente, se dirá que θ^1 ϵ-domina a θ^2 (denotado como $\theta^1 \preceq_\epsilon \theta^2$) para un determinado $\epsilon > 0$ si*

$$\forall i \in B := [1, \ldots, s], \quad J_i(\theta^1) - \epsilon \ \leq \ J_i(\theta^2). \tag{4.1}$$

$\triangle$

La figura 4.1 muestra gráficamente (para dos dimensiones) el concepto ϵ-dominante comparado con el de Pareto dominante definido en el capítulo anterior (ver definición 3.2). Se deduce también, por simple inspección gráfica que si un individuo domina a otro también lo ϵ-dominará.

A partir de la definición ϵ-dominante es posible pasar a definir un conjunto óptimo ϵ-Pareto, como aproximación discretizada del conjunto de óptimos de Pareto Θ_P de la siguiente manera.

Definición 4.2 (Conjunto óptimo ϵ-Pareto): *Dado un conjunto de vectores Θ en el espacio de soluciones D y un $\epsilon > 0$, el conjunto Θ_{P_ϵ} será un conjunto óptimo ϵ-Pareto de Θ si y sólo si*

[3]Existen muchas aproximaciones diferentes en la literatura, en [70] se muestra una revisión de las mismas.

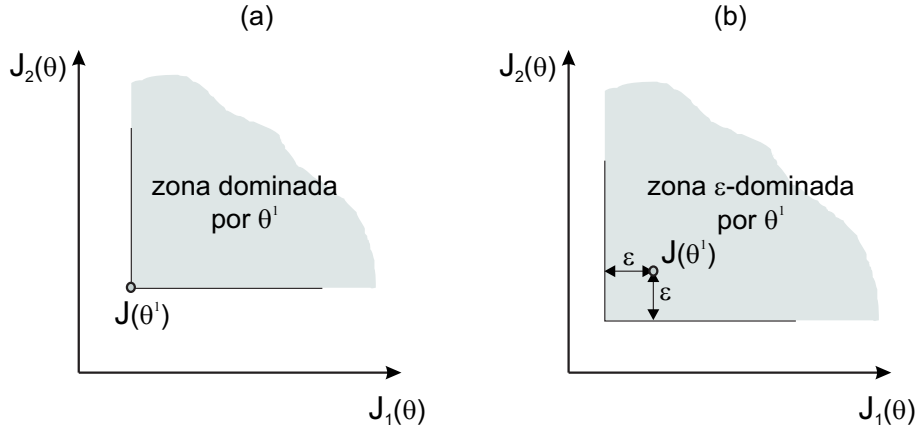

Figura 4.1: Representación gráfica de los conceptos (a) Pareto dominante y (b) ϵ-dominante.

1. *Cualquier vector en* Θ *es* ϵ*-dominado por algún vector en* Θ_{P_ϵ}*, es decir:*

$$\forall \theta \in \Theta, \ \ \exists \theta^* \in \Theta_{P_\epsilon} : \theta^* \preceq_\epsilon \theta.$$

2. *Los vectores* $\theta^* \in \Theta_{P_\epsilon}$ *son no dominados de* Θ*, es decir:*

$$\nexists \ \theta \in \Theta : \theta \preceq \theta^*.$$

$\triangle$

De la definición anterior se deduce que:

- Dado un conjunto Θ, Θ_{P_ϵ} no tiene porque ser un conjunto único. La figura 4.2 pone de manifiesto este hecho donde a partir del mismo conjunto $\mathbf{J}(\Theta)$ se dispone de dos conjuntos $\mathbf{J}(\Theta_{P_\epsilon})$ diferentes. Esto ocurre cuando varias soluciones no dominadas se ϵ-dominan entre ellas.
- Al sólo poder existir vectores no dominados en Θ_{P_ϵ} se asegura el no deterioro del algoritmo y la convergencia hacia Θ_P.

Definición 4.3 (Conjunto $\mathcal{P}_\epsilon(\Theta)$): *Al conjunto de todos los conjuntos óptimos* ϵ*-Pareto se le denominará como* $\mathcal{P}_\epsilon(\Theta)$*.*

$\triangle$

Definición 4.4 (Frente ϵ-Pareto): *Dado un conjunto óptimo* ϵ*-Pareto* Θ_{P_ϵ}*, se define el frente* ϵ*-Pareto como:*

$$\mathbf{J}(\Theta_{P_\epsilon}) := \{\mathbf{J}(\theta) | \ \theta \in \Theta_{P_\epsilon}\},$$

es decir, la imagen de Θ_{P_ϵ} *en el espacio de funciones de fitness.*

$\triangle$

El objetivo que se persigue con la definición del conjunto óptimo ϵ-Pareto es la determinación de una aproximación discreta del frente de Pareto (cuanto más cercana y

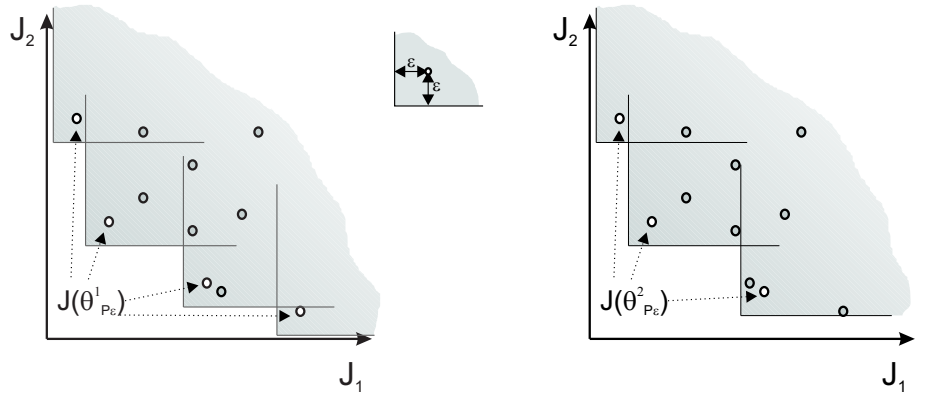

Figura 4.2: Representación gráfica de dos conjuntos óptimos ϵ-Pareto diferentes $\Theta^1_{P_\epsilon}$ y $\Theta^2_{P_\epsilon}$ (representados en el espacio de funciones $\mathbf{J}(\cdot)$ mediante círculos blancos) para un mismo conjunto Θ (los círculos representan $\mathbf{J}(\Theta)$). Evidentemente una tercera posibilidad sería el conjunto $\Theta^3_{P_\epsilon} = \Theta^1_{P_\epsilon} \cup \Theta^2_{P_\epsilon}$.

mejor distribuida mejor) que presente un tamaño acotado de puntos para que pueda ser utilizada, más fácilmente, en la determinación de la solución definitiva (*decision making*).

El tamaño acotado del conjunto óptimo ϵ-Pareto y el no deterioro del algoritmo se puede conseguir [49], por ejemplo, dividiendo el espacio de las funciones de *fitness* mediante un *grid* con coordenadas 0, ϵ_i, $2\epsilon_i$, $3\epsilon_i$, ... para cada una de las dimensiones[4] $i \in [1 \dots s]$ y obligando a que sólo pueda existir, en el archivo $A(t)$, una solución en un mismo *box* (del *grid*) siendo dicho *box* no dominado (ver figura 4.3).

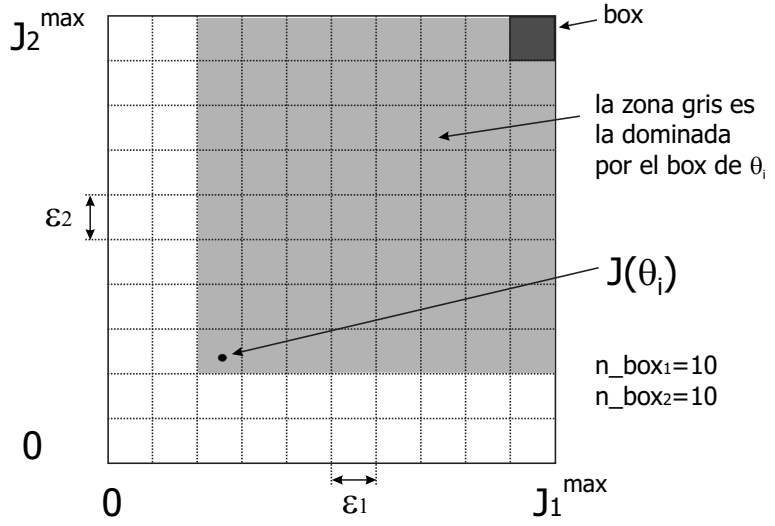

Figura 4.3: Representación gráfica del *grid*, *box* y dominancia del *box* de $\mathbf{J}(\theta^i)$, para un caso bidimensional.

[4]Con el objetivo de generalizar el problema, se plantea la posibilidad de que existan valores de ϵ diferentes para cada dimensión.

Para poder demostrar esta cuestión es necesario establecer una serie de definiciones previas.

Definición 4.5 (Box): *Dado un vector θ en el espacio de soluciones, cuya imagen en el espacio de las funciones de fitness es $\mathbf{J}(\theta)$, se define su box para valores $\epsilon_i > 0$ como el vector $\mathbf{box}(\theta) = [box_1(\theta) \dots box_s(\theta)]$ donde:*

$$box_i(\theta) = \left\lfloor \frac{J_i(\theta)}{\epsilon_i} \right\rfloor \ \forall i \in [1 \dots s].$$

$\triangle$

Esto hace que $box_i(\theta) \in [0 \dots (n\_box_i - 1)]$, siendo n_box_i el número de divisiones del *grid* en la dimensión i

$$n\_box_i = \left\lceil \frac{J_i^{max}}{\epsilon_i} \right\rceil,$$

donde $0 \leq J_i(\theta) \leq J_i^{max}$ y $J_i^{max} \geq \epsilon_i$.

Definición 4.6 (Box dominante): *Dado dos vectores θ^1, θ^2 en el espacio de soluciones, cuyos boxes son $\mathbf{box}(\theta^1)$ y $\mathbf{box}(\theta^2)$ respectivamente, se dirá que θ^1 box domina a θ^2, $\mathbf{box}(\theta^1) \preceq \mathbf{box}(\theta^2)$, si y sólo si*

$$\forall i \in B := [1, \dots, s], box_i(\theta^1) \leq box_i(\theta^2) \wedge \exists i \in B : box_i(\theta^1) < box_i(\theta^2).$$

$\triangle$

Teorema 4.1: *Si un vector θ^1 box domina a otro vector θ^2, $\mathbf{box}(\theta^1) \preceq \mathbf{box}(\theta^2)$, también lo ϵ-dominará, $\theta^1 \preceq_\epsilon \theta^2$.*

Demostración. *Si $\mathbf{box}(\theta^1) \preceq \mathbf{box}(\theta^2)$, al menos existirá un $i \in [1 \dots s] : box_i(\theta^1) < box_i(\theta^2)$ pudiendo darse la situación en la que para algunas dimensiones, $box_i(\theta^1) = box_i(\theta^2)$. Si*

$$box_i(\theta^1) < box_i(\theta^2) \Rightarrow \left\lfloor \frac{J_i(\theta^1)}{\epsilon_i} \right\rfloor < \left\lfloor \frac{J_i(\theta^2)}{\epsilon_i} \right\rfloor \Rightarrow$$

$$J_i(\theta^1) < J_i(\theta^2) \Rightarrow J_i(\theta^1) - \epsilon_i < J_i(\theta^2)$$

condición, esta última, que se ha de cumplir para que $\theta^1 \preceq_\epsilon \theta^2$ (ver definición 4.1). Por otra parte si

$$box_i(\theta^1) = box_i(\theta^2) \Rightarrow box_i(\theta^1) - 1 < box_i(\theta^2) \Rightarrow$$

$$\left\lfloor \frac{J_i(\theta^1)}{\epsilon_i} \right\rfloor - 1 < \left\lfloor \frac{J_i(\theta^2)}{\epsilon_i} \right\rfloor \Rightarrow \left\lfloor \frac{J_i(\theta^1)}{\epsilon_i} - 1 \right\rfloor < \left\lfloor \frac{J_i(\theta^2)}{\epsilon_i} \right\rfloor$$

$$\frac{J_i(\theta^1)}{\epsilon_i} - 1 < \frac{J_i(\theta^2)}{\epsilon_i} \Rightarrow J_i(\theta^1) - \epsilon_i < J_i(\theta^2)$$

con lo cual se confirma, para cualquier caso, que $\theta^1 \preceq_\epsilon \theta^2$.

$\Diamond$

Con estas definiciones y demostraciones es posible establecer, más formalmente, la siguiente definición del procedimiento que gestionará el contenido del archivo $A(t)$ donde se almacenará la solución Θ_{P_ϵ} al problema de optimización.

Definición 4.7 (Inclusión de θ en $A(t)$): *Dado un vector de soluciones θ y el archivo $A(t)$, θ será incluido en el archivo si y sólo si $\nexists \theta^* \in A(t)$ tal que*

$$\mathbf{box}(\theta^*) \prec \mathbf{box}(\theta) \tag{4.2}$$

$$\vee$$

$$(\mathbf{box}(\theta^*) = \mathbf{box}(\theta)) \wedge (\theta^* \prec \theta). \tag{4.3}$$

Al mismo tiempo, si θ es incluido en el archivo serán eliminados de $A(t)$ todos las soluciones θ^* que hacen cierta la siguiente condición

$$\mathbf{box}(\theta) \prec \mathbf{box}(\theta^*) \tag{4.4}$$

$$\vee$$

$$(\mathbf{box}(\theta) = \mathbf{box}(\theta^*)) \wedge (\theta \prec \theta^*). \tag{4.5}$$

$\triangle$

Gracias a las ecuaciones (4.2) y (4.4) siempre se mantendrán ocupados *boxes* que sean no dominados y dentro de los *boxes* los individuos serán no dominados (ecuaciones (4.3) y (4.5)) garantizándose, como más tarde se demostrará, que $A(t)$ será un conjunto óptimo ϵ-Pareto, y por lo tanto, se garantiza la convergencia del algoritmo.

Utilizando el procedimiento de actualización anterior es posible establecer los siguientes teoremas y sus demostraciones.

Teorema 4.2: *Para cualquier conjunto Θ en el espacio de soluciones, que representa el conjunto de todos los individuos creados durante el proceso de optimización y cuyo espacio de funciones se encuentra dividido por un grid con anchuras $\epsilon_i > 0$, es posible dar con un conjunto óptimo ϵ-Pareto, cuyos individuos son box no dominados, con tamaño $|\Theta_{P_\epsilon}|$ acotado*[5]

$$|\Theta_{P_\epsilon}| \leq \frac{\prod_{i=1}^{s} n\_box_i}{n\_box_{max}}, \tag{4.6}$$

[5]La cota establecida por la ecuación (4.6) es inferior o igual a la planteada en [105] $\prod_{i=1}^{s-1} n\_box_i$.

donde $n\_box_{max} = \max_i n\_box_i$.

Demostración. *El espacio de funciones está dividido por un grid que contiene*

$$\prod_{i=1}^{s} n\_box_i$$

boxes, en los que sólo puede existir una solución. Si se agrupan estos boxes en grupos, de tamaño máximo, donde sólo varíe una de sus coordenadas, se dispondrá de $\frac{\prod_{i=1}^{s} n\_box_i}{n\_box_{max}}$ *grupos diferentes con* n_box_{max} *boxes cada uno de ellos. Dado que sólo uno de los boxes de cada grupo (es decir, el individuo que ocupe dicho box) puede pertenecer a* Θ_{P_ϵ}*, por la propia definición de box dominancia, el número máximo de individuos en* Θ_{P_ϵ} *será* $\frac{\prod_{i=1}^{s} n\_box_i}{n\_box_{max}}$.

$\Diamond$

Teorema 4.3: *Para cualquier conjunto* Θ *en el espacio de soluciones, cuyo espacio de funciones se encuentra dividido por un grid con anchuras* $\epsilon_i > 0$*, el contenido del archivo* $A(t)$ *resultante de la inclusión de los individuos de* Θ *sobre él, mediante el procedimiento descrito en la definición 4.7, será un conjunto óptimo* ϵ*-Pareto*

$$A(t) \in \mathcal{P}_\epsilon(\Theta).$$

Demostración. *Se asume inicialmente que* $A(t) \notin \mathcal{P}_\epsilon(\Theta)$*. Para ello, según la definición 4.2, se debe cumplir cualquiera de las dos siguientes condiciones:*

1. *Existe un individuo en* Θ *que no es* ϵ*-dominado por ninguno de los individuos en* $A(t)$*, es decir,*
$$\exists \theta \in \Theta : \forall \theta' \in A(t),\ \theta' \not\preceq_\epsilon \theta.$$

2. *Existe un individuo en* $A(t)$ *que es dominado por algún individuo en* Θ*, es decir,*
$$\exists \theta \in \Theta, \exists \theta' \in A(t) : \theta \preceq \theta'.$$

La ***condición 1*** *no se cumple ya que la ecuación (4.2) asegura que si un individuo no es incluido en el archivo es porque existe en él un individuo que lo* ϵ*-domina y si este individuo está en el archivo y es posteriormente excluido de él, es porque hay un individuo que, al intentar ser introducido en el archivo, lo* ϵ*-domina (ecuación (4.4)).*

La ***condición 2*** *tampoco se cumple, pues la ecuación (4.4) asegura que los individuos que entren en el archivo sean no dominados y la ecuación (4.5) se encarga de eliminar aquellos que, estando en el archivo, pasen a ser dominados.*

Por lo tanto, asumir que $A(t) \notin \mathcal{P}_\epsilon(\Theta)$ *resulta falso.*

$\Diamond$

La determinación de los coeficientes ϵ_i será dependiente del problema en cuestión y condicionará el tamaño máximo de individuos en el Θ_{P_ϵ} que caracterizarán el frente de Pareto. Evidentemente el diseñador debería ajustar dichos coeficientes en función del significado físico de las funciones de *fitness*, ya que, en cierta manera, condicionarán la distancia entre puntos del frente (dado que no pueden coexistir dos puntos en el mismo *box*) creando un frente discretizado o también conocido como *Smart Pareto Front* [115].

Existen otras formas diferentes de crear el *grid* a la planteada hasta ahora. Por ejemplo, dividir el espacio de las funciones de *fitness* mediante un *grid* con coordenadas 0, ϵ_i, ϵ_i^2, $\epsilon_i^3 \ldots$, siendo $\epsilon_i > 1 \ \forall i \in [1 \ldots s]$. De manera que, ahora el *box* se calcularía mediante la expresión

$$box_i(\theta) = \left\lfloor \frac{\log(J_i(\theta))}{\log(\epsilon_i)} \right\rfloor \ \forall i \in [1 \ldots s]$$

y el número de *boxes* en cada dimensión sería

$$n\_box_i = \left\lceil \frac{\log(J_i^{max})}{\log(\epsilon_i)} \right\rceil,$$

siendo válida la expresión (4.6) para determinar el tamaño máximo de Θ_{P_ϵ}, incorporando las nuevas expresiones.

4.3. Descripción de ϵ-MOGA

En este apartado se va a presentar el algoritmo ϵ-MOGA, desarrollado a partir del algoritmo presentado en [45] y utilizando las directrices dadas en [105, 104] con el objetivo de dar con un conjunto óptimo ϵ-Pareto, Θ_{P_ϵ}.

Las características del algoritmo son las siguientes:

- Se trata de un algoritmo elitista (como posteriormente se demostrará) con un archivo que almacena los individuos ϵ-dominantes y una población auxiliar adicional (no utilizada en [45]).
- No generacional, solapable y preservativo.
- Utiliza codificación real de los parámetros y operadores de cruce (recombinación intermedia extendida) y mutación (aleatoria con distribución *gaussiana*) con ajuste de parámetros no adaptativo basado en funciones[6].
- La selección se resuelve aleatoriamente[7] y en la determinación se utilizan conjuntamente torneo y estado uniforme.

[6] En [45] se utiliza sólo el operador cruce por simulación binaria (SBX).
[7] En [45] se mezcla la selección por torneo binario y la aleatoria.

- La implementación utiliza una arquitectura paralela maestro-esclavo y se ejecuta en una plataforma multiordenador compuesta de n ordenadores. El algoritmo está implementado en código C para multiplataforma utilizando librerías GPL[8].

El algoritmo está compuesto de tres poblaciones (ver figura 4.4).

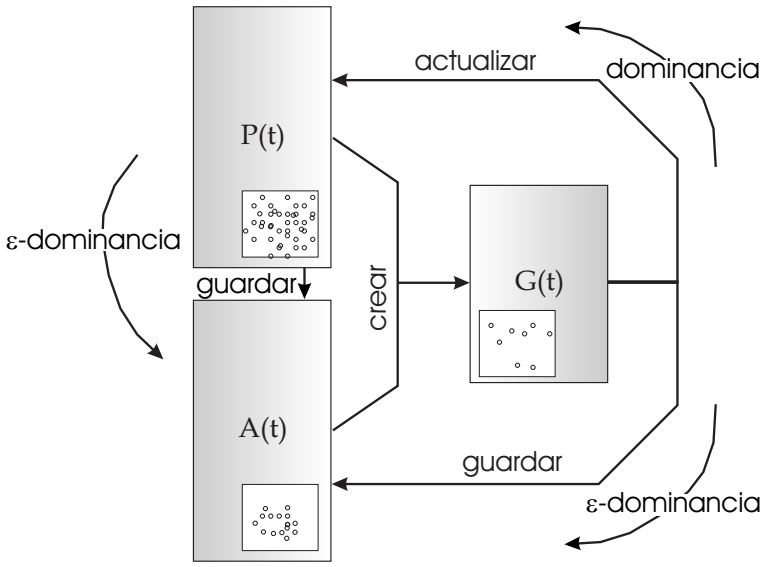

Figura 4.4: Estructura del algoritmo ϵ-MOGA.

1. $P(t)$ es la población principal y se encarga de explorar el espacio de búsqueda D a lo largo de las iteraciones del algoritmo (t). El tamaño de la población es fijo e igual a $Nind_P$.

2. $A(t)$ es el archivo donde se almacena la solución al problema de optimización, es decir, la aproximación al frente de Pareto Θ_{P_ϵ}. El tamaño de esta población $Nind_A$ es variable pero como se demostró en la sección anterior su tamaño permanecerá acotado (ecuación 4.6). De esta manera, se puede establecer la memoria necesaria para almacenar el contenido del archivo asegurando que ésta no será desbordada.

3. $G(t)$ es una población auxiliar que se utiliza para almacenar los individuos nuevos que el algoritmo genera en cada iteración. El tamaño de esta población es $Nind_G$ y tal y como el algoritmo ha sido diseñado $Nind_G$ debe ser un número par.

El pseudocódigo del algoritmo ϵ-MOGA es el siguiente:

```
1. t:=0
2. A(t):=∅
3. P(t):=ini_random(D)
```

[8]GPL es el acrónimo de *General Public License* (http://es.gnu.org/licencias/).

```
4. eval(P(t))
5. A(t):=guardar(P(t),A(t))
6. while t<t_max do
7.      G(t):=crear(P(t),A(t))
8.      eval(G(t))
9.      A(t+1):=guardar(G(t),A(t))
10.     P(t+1):=actualizar(G(t),P(t))
11.     t:=t+1
12. end while
```

Algoritmo 4.1: Algoritmo ϵ-MOGA.

A continuación, se detallan los pasos más importantes del algoritmo:

- **Línea 3**. La población $P(0)$ es inicializada con $Nind_P$ individuos creados aleatoriamente dentro del espacio de búsqueda D.
- **Líneas 4 y 8**. La función *eval* calcula el vector de las funciones de *fitness* $\mathbf{J}(\theta)$ para cada individuo θ de $P(t)$ (línea 4) o de $G(t)$ (línea 8). La carga computacional de esta función se reparte entre los 4 ordenadores de la plataforma multiordenador reduciendo así el tiempo de cómputo considerablemente, ya que normalmente, la mayor carga computacional recae sobre la evaluación de las funciones de *fitness*.
- **Líneas 5 y 9**. La función *guardar* analiza uno a uno los individuos de $P(t)$ (línea 5) o de $G(t)$ (línea 9) para determinar si serán incluidos en $A(t)$, para ello, el individuo tendrá que cumplir la condición de inclusión de la definición 4.7 eliminando los individuos que la misma definición especifica. En el caso de la línea 5, $\Theta = P(0)$ y para el caso de la línea de código 9

$$\Theta = P(0) \bigcup \left(\bigcup_{0 \leq \tau < t} G(t) \right).$$

  es decir, todos los individuos creados hasta el instante t.

  Si al intentar introducir un individuo θ^1 en el archivo, su *box* ($\mathbf{box}(\theta^1)$) está ocupado por otro individuo θ^2 del archivo, es decir $\mathbf{box}(\theta^1) = \mathbf{box}(\theta^2)$, y ninguno de ellos domina al otro, se dejará en el archivo aquel individuo que más cerca esté de las coordenadas del *box* que ocupan[9]. De esta manera se cumple la condición de que cada *box* sólo puede ser ocupado por un individuo al mismo tiempo, y por otra parte, favorece una mejor distribución del frente ϵ-Pareto (ver figura 4.5).
- **Línea 7**. La función *crear* es la encargada de generar nuevos individuos y almacenarlos en la población $G(t)$ en cada iteración usando el siguiente procedimiento:

[9]Si esta situación es tratada estrictamente mediante la definición 4.7 conllevaría a la no inclusión del individuo θ^1. La coordenada del *box* se obtiene como $[n\_box_1 \cdot \epsilon_1 \ldots n\_box_s \cdot \epsilon_s]$, es decir, el vértice inferior izquierdo del *box* en el caso bidimensional.

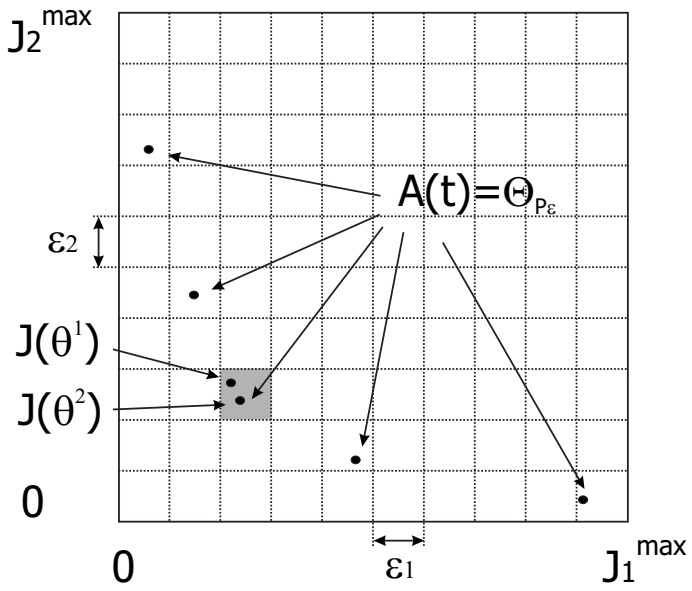

Figura 4.5: Ilustración de un posible contenido del archivo $A(t)$. Se plantea la situación donde se analiza si θ^1 debe ser incorporado al archivo. Ocupa el mismo *box* que θ^2 (el *box* $[2,2]$ en concreto) y ambos son individuos no dominados, sin embargo, θ^1 no es incluido en el archivo (en cuyo caso reemplazaría a θ^2) ya que $|\mathbf{J}(\theta^1) - [2,2]| > |\mathbf{J}(\theta^2) - [2,2]|$.

1. Se seleccionan dos individuos aleatoriamente, θ^{p1} de $P(t)$, y θ^{p2} de $A(t)$.
2. Se lanza un número aleatorio $u \in [0 \dots 1]$. Si $u > P_{c/m}$ (probabilidad de cruce mutación) se realiza el paso 3 (cruce) sino el paso 4 (mutación).
3. θ^{p1} y θ^{p2} son cruzados mediante la técnica de recombinación lineal extendida generando dos individuos nuevos θ^{h1} y θ^{h2} de la siguiente manera:

$$\theta_i^{h1} = \alpha_i(t) \cdot \theta_i^{p1} + (1 - \alpha_i(t)) \cdot \theta_i^{p2},$$

$$\theta_i^{h2} = (1 - \alpha_i(t)) \cdot \theta_i^{p1} + \alpha_i(t) \cdot \theta_i^{p2}.$$

El parámetro $\alpha_i(t)$ es un valor aleatorio con distribución uniforme $\in [-d(t), 1+ d(t)]$ y $d(t)$ a su vez es un parámetro que se ajusta utilizando una función exponencial decreciente como en *simulated annealing*[10].

$$d(t) = \frac{d_{ini}}{\sqrt{1 + \left(\left(\frac{d_{ini}}{d_{fin}}\right)^2 - 1\right) \frac{t}{(t_{max}-1)}}}, \quad d_{ini} > d_{fin}. \tag{4.7}$$

La figura 4.6 muestra la evolución exponencial decreciente de $d(t)$ desde $t = 0$, donde $d(t) = d_{ini}$, hasta $t = t_{max} - 1$ para el cual $d(t) = d_{fin}$.

[10]El objetivo que se persigue con este tipo de expresiones es potenciar la exploración sobre el espacio de búsqueda en las iteraciones iniciales del algoritmo e ir reduciéndola para potenciar la convergencia, a medida que el algoritmo va evolucionando.

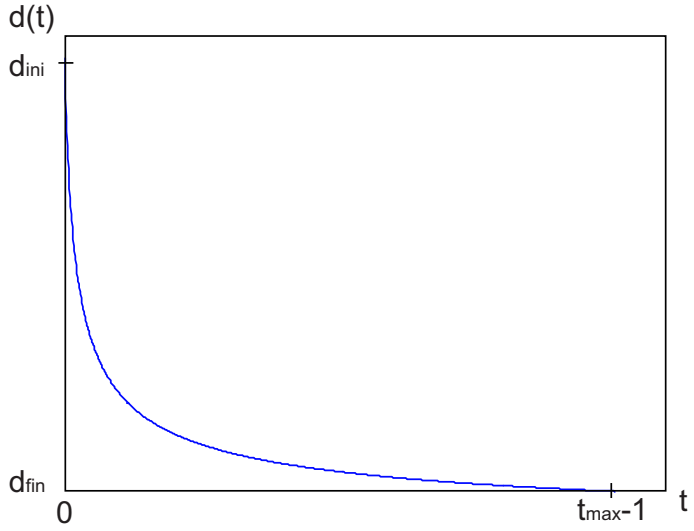

Figura 4.6: Evolución de $d(t)$ desde $t = 0$ hasta $t = t_{max} - 1$. Los parámetros d_{ini} y d_{fin} determinan los valores iniciales y finales de la función (4.7).

4. θ^{p1} y θ^{p2} son mutados utilizando mutación aleatoria con distribución *gaussiana*.

$$\theta_i^{h1} = \theta_i^{p1} + N(0, \beta 1_i(t))$$
$$\theta_i^{h2} = \theta_i^{p2} + N(0, \beta 2_i(t)),$$

donde las varianzas $\beta 1_i(t)$ y $\beta 2_i(t)$ se miden en porcentaje de la cantidad $(\theta_{i\,max} - \theta_{i\,min})$ y se determinan utilizando una función análoga a la utilizada para el parámetro $d(t)$ anterior.

$$\beta(t) = \frac{\beta_{ini}}{\sqrt{1 + \left(\left(\frac{\beta_{ini}}{\beta_{fin}}\right)^2 - 1\right)\frac{t}{(t_{max}-1)}}}.$$

5. θ^{h1} y θ^{h2} son incluidos en $G(t)$.

Este procedimiento se repite $Nind_G/2$ veces hasta que $G(t)$ esté llena.

- **Línea 10**. La función **actualizar** determina qué individuos de $G(t)$ serán incluidos en $P(t)$. Cada individuo θ^G de $G(t)$ sustituirá a un individuo θ^P elegido aleatoriamente de entre los individuos de $P(t)$ que son dominados por θ^G. Si en $P(t)$ no existe ningún individuo dominado por θ^G, éste no será incluido.

Finalmente, cuando $t = t_{max}$, los individuos contenidos en el archivo $A(t)$ constituirán la solución al problema de optimización multiobjetivo Θ_{P_ϵ}, considerándose Θ como el conjunto de individuos generados por las líneas de código 3 y 7, es decir,

$$\Theta = P(0) \bigcup \left(\bigcup_{0 \leq \tau < t_{max}-1} G(t) \right).$$

4.4. Evaluación del algoritmo

En esta sección, se pretende evaluar las prestaciones del algoritmo ϵ-MOGA utilizando una serie de problemas de optimización diseñados expresamente para evaluar algoritmos de optimización multiobjetivo [36]. En el anexo A.1 se encuentran las funciones correspondientes a los cinco problemas que se utilizarán, así como el conjunto de óptimos de Pareto de cada uno de ellos con sus correspondientes frentes de Pareto.

Para cada uno de los problemas se comparan los resultados obtenidos por el algoritmo ϵ-MOGA, el algoritmo ϵ-MOEA presentado en [45] y los algoritmos de búsqueda aleatoria y exhaustiva[11]. Los indicadores que se utilizan para analizar y comparar los resultados son: número de soluciones encontradas, distancia generacional, espaciado, individuos no dominados, ratio del hiper-área, ratio *box* del frente[12] y alcance comparativo (ver sección 3.5).

Dado que el proceso de optimización incorpora variables aleatorias, la optimización de cada problema, se repite 10 veces utilizando semillas diferentes[13]. De esta manera, si se median los resultados es posible presentar conclusiones menos dependientes de la secuencia de números aleatorios utilizada en una optimización en concreto. Por lo tanto, se mostrarán los valores máximos, mínimos, la media y la varianza de cada indicador para las 10 optimizaciones diferentes que se lleven a cabo.

Para realizar la evaluación se han ajustado los siguientes parámetros de configuración del algoritmo ϵ-MOGA que, se mantienen constantes para cada una de los problemas:

- $Nind_G = 4$, de esta manera, a la hora de ejecutar la línea 8 del código del algoritmo ϵ-MOGA, se distribuye el coste computacional entre los 4 ordenadores que conforman la plataforma multiordenador utilizada.
- $Nind_P = 100$, como sugieren otros autores que han utilizado los mismos problemas [43, 152].
- t_{max} se elegirá en cada problema en concreto dependiendo de la dimensión y tamaño del espacio de soluciones.

[11] No se compara ϵ-MOGA directamente con otros MOEAs, ya que, ϵ-MOEA ha sido comparado con otros algoritmos como NSGA-II, PESA y SPEA2 [45] al comparar ϵ-MOGA con ϵ-MOEA también se está comparando ϵ-MOGA, de forma indirecta, con el resto de algoritmos.

[12] Es el área o volumen del *box* formado por los extremos del frente de Pareto determinado por el algoritmo en cuestión, dividido por el área o volumen del *box* formado por los extremos del frente de Pareto real.

[13] Excepto para la búsqueda exhaustiva que es determinística.

- $P_{c/m} = 0.1$ para que, en media, se crucen el 90 % de individuos y por lo tanto se muten el 10 % y sea comparable a lo que proponen otros autores, que sugieren $P_c = 0.9$.
- $d_{ini} = 0.25$, $d_{fin} = 0.1$, $\beta_{ini} = 10.0$ y $\beta_{fin} = 0.1$ se escogen con estos valores para favorecer una buena exploración del espacio de búsqueda en las primeras iteraciones y una posterior convergencia.

Para el algoritmo ϵ-MOEA los parámetros utilizados son:

- $Nind_G = 1$ como se propone en [45].
- $Nind_P = 100$ para coincida con la población del algoritmo ϵ-MOGA.
- t_{max} se elegirá para que las funciones de test se evalúen el mismo número de veces que con el algoritmo ϵ-MOGA.
- $\eta_c = 15$ para el cruce SBX con $P_c = 1$ y sin utilizar mutación, como se sugiere en [45].

Para la búsqueda aleatoria, el planteamiento a seguir es el siguiente:

1. Se crea una población de N individuos sobre el espacio de búsqueda de forma aleatoria. N se escoge de manera que se iguale a los algoritmos anteriores en número de evaluaciones de las funciones de *fitness*.
2. Se selecciona los individuos no dominados de dicha población como solución al problema de optimización.

Para la búsqueda exhaustiva el planteamiento que se sigue es el siguiente:

1. Se crea una población de aproximadamente N individuos[14] sobre el espacio de búsqueda colocándolos de forma uniforme utilizando un *grid* regular. De esta manera, la ubicación de los individuos es totalmente determinística.
2. Se selecciona los individuos no dominados de dicha población como solución al problema de optimización.

A continuación, se muestran los resultados obtenidos para cada problema con los procedimientos descritos anteriormente.

- **Problema MOP1**. Para este problema, en concreto, se utilizan los siguientes parámetros para los algoritmos ϵ-MOGA y ϵ-MOEA:

[14]Cada dimensión del espacio de búsqueda $D \subseteq \mathcal{R}^L$ se divide en q partes, de esta manera el número total de *boxes* del *grid* será $N = q^L$. Dado que tanto L como q son números enteros se escogerá q para que $q^L = N \approxeq$ al número evaluaciones de las funciones de *fitness* que realicen los otros algoritmos.

- $J_1^{max} = 4$ y $J_2^{max} = 4$[15].
- $\epsilon_1 = 0.08$ y $\epsilon_2 = 0.08$[16].
- $t_{max} = 4975$ para el algoritmo ϵ-MOGA y $t_{max} = 19900$ para el algoritmo ϵ-MOEA de manera que se evalúen 20000 veces las funciones de test[17], como se sugiere en [43, 152].

Por lo tanto $N = 20000$ para el algoritmo de búsqueda aleatoria y para de búsqueda exhaustiva dado que el espacio de soluciones sólo tiene una dimensión se escoge $q = 20000$ y adicionalmente se repite la optimización con $q = 20001$ y $q = 19900$ por no disponer de tan sólo una optimización.

La tabla 4.1 muestra los resultados de la optimización del problema MOP1 con los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, respectivamente. Sobre la tabla aparecen en negrita los mejores resultados obtenidos, en media, para cada indicador. Del análisis de los resultados se pueden obtener las siguientes conclusiones:

- Los algoritmos ϵ-MOGA y ϵ-MOEA son claramente superiores a los algoritmos de búsqueda exhaustiva y aleatoria, para un mismo coste computacional (20000 evaluaciones de las funciones de *fitness*). El motivo principal de los malos resultados obtenidos por la búsqueda aleatoria y exhaustiva son debidos a que el MOP1 presenta un espacio de soluciones muy grande, del orden de $2 \cdot 10^5$ (aunque sólo sea de una dimensión).
- Los resultados obtenidos por los algoritmos ϵ-MOGA y ϵ-MOEA son prácticamente iguales y las diferencias, aunque existen, son tan pequeñas que no merece la pena mencionarlas.
- Tanto ϵ-MOGA como ϵ-MOEA han conseguido aproximar el frente de Pareto con $|\Theta_{P_\epsilon}| = 25$ soluciones[18] todas ellas pertenecientes al frente de Pareto real ($IND = 25$ indica que ninguna de las 25 soluciones es dominada por las soluciones óptimas de Pareto Θ_P^{MOP1}) de ahí que la convergencia del algoritmo haya sido perfecta. $SP \simeq 0$ denota una buena distribución de las soluciones sobre el frente de Pareto.

La figura 4.7 muestra los frentes obtenidos por los 4 algoritmos[19]. Si se observa esta figura se puede detectar que las soluciones de los algoritmos ϵ-MOGA y ϵ-MOEA no han capturado los extremos del frente, como pone de manifiesto el indicador

[15]Estos valores que se utilizan en la determinación de la cota de $|\Theta_{P_\epsilon}|$ pueden ser escogidos gracias a que se conoce el frente de Pareto solución al problema de optimización.

[16]Estos valores han sido escogidos para que sean lo suficientemente pequeños como para poder caracterizar con precisión el frente de Pareto.

[17]El número de evaluaciones se determina como $Nind_P + Nind_G * t_{max}$.

[18]Los resultados ideales serían $|\Theta_{P_\epsilon}| \uparrow\uparrow$, IND=$|\Theta_{P_\epsilon}|$, DG=0, HR=1, SP=0, y RBOX=1.

[19]De las diferentes optimizaciones llevadas a cabo para cada uno de los algoritmos se escoge, para ser representada gráficamente, aquella que más cerca está de los valores medios obtenidos.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	HR	SP	RBOX
ϵ-MOGA						
mínimo	25	25	0.002888	0.92899	0.0076606	0.51437
máximo	25	25	0.0029556	0.92902	0.0076937	0.51469
media	**25**	**25**	**0.0029216**	**0.92901**	0.0076754	**0.5145**
std	0	0	1.966e-005	7.0709e-006	1.0827e-005	8.7248e-005
ϵ-MOEA						
mínimo	25	25	0.0028875	0.92899	0.0076317	0.51433
máximo	25	25	0.0031832	0.92904	0.0076709	0.51467
media	**25**	**25**	0.0029616	**0.92901**	**0.0076597**	0.51439
std	0	0	8.9784e-005	1.5786e-005	1.2341e-005	0.00010143
B. Aleatoria						
mínimo	1	0	0.0067227	0.063353	0	0
máximo	34	1	5.6953	20.892	0.10071	4.577
media	23.8	0.1	4.9636	18.256	0.050998	4.021
std	8.7407	0.31623	1.7446	6.3995	0.037081	1.4143
B. Exhaustiva						
mínimo	1	0	0	0	0	0
máximo	1	1	22.844	84.775	0	0
media	1	0.33333	7.6148	28.258	0	0
std	0	0.5770	13.189	48.945	0	0

Tabla 4.1: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$ (número de soluciones), IND (individuos no dominados), DG (distancia generacional), HR (ratio de hiper-área), SP (espaciado) y RBOX (ratio *box* del frente) resultado de la optimización del problema MOP1 con ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

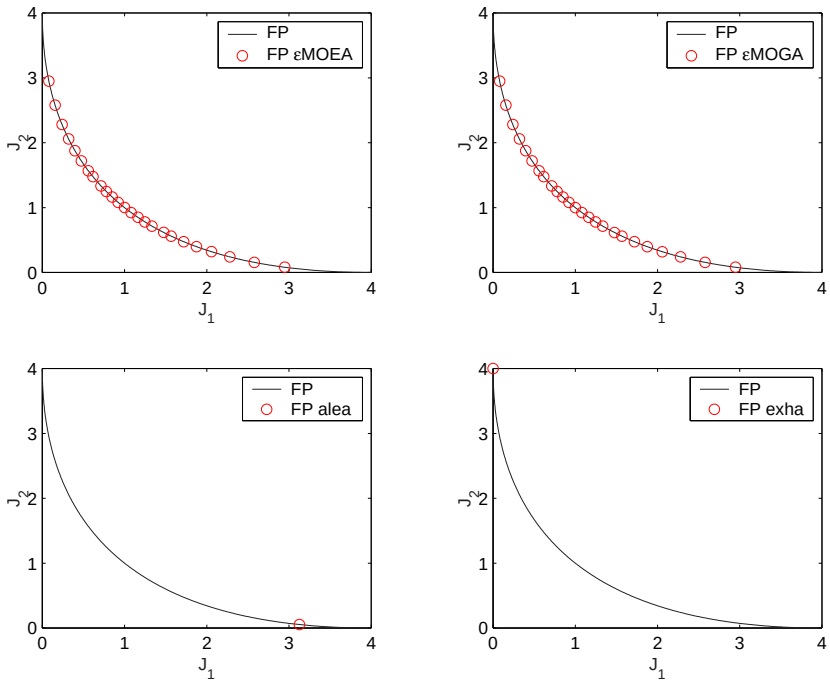

Figura 4.7: Frentes de Pareto obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva para MOP1.

$RBOX \simeq 0.5$ ya que lo ideal es que esté próximo a la unidad. Se podría caracterizar mejor el frente de Pareto reduciendo el valor de ϵ.

- **Problema MOP2**. Para este problema, en concreto, se utilizan los siguientes parámetros para los algoritmos ϵ-MOGA y ϵ-MOEA:

  - $J_1^{max} = 1$ y $J_2^{max} = 1$.
  - $\epsilon_1 = 0.02$ y $\epsilon_2 = 0.02$.
  - $t_{max} = 4975$ para el algoritmo ϵ-MOGA y $t_{max} = 19900$ para el algoritmo ϵ-MOEA de manera que se evalúen 20000 veces las funciones de test.

  Por lo tanto, de nuevo $N = 20000$ para el algoritmo de búsqueda aleatoria y para el de búsqueda exhaustiva dado que el espacio de soluciones en este caso es de dimensión $\mathcal{R}^3$ se han elegido dos valores para el parámetro q, $q = 27$ por lo tanto $27^3 = 19683$ y $q = 28$ por lo tanto $28^3 = 21952$.

  La tabla 4.2 muestra los resultados de la optimización del problema MOP2 con los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, respectivamente. Del análisis de los resultados se pueden obtener las siguientes conclusiones:

  - Los algoritmos ϵ-MOGA y ϵ-MOEA son superiores a los algoritmos de búsqueda exhaustiva y aleatoria como en el caso anterior.
  - Con ϵ-MOGA se han conseguido 42 soluciones pertenecientes al frente (excepto en una optimización que ha conseguido 41) con lo que se puede concluir que la convergencia del algoritmo ha sido perfecta, además un $DG \simeq 0$ y un $HR \simeq 1$ así lo corroboran, mientras que con ϵ-MOEA, en ocasiones, no ha sido así. De nuevo $SP \simeq 0$ denota una buena distribución de las soluciones sobre el frente de Pareto, ahora $RBOX \simeq 1$ ya que, los extremos del frente de Pareto se han caracterizado mejor que en el problema MOP1. Si se analiza $RBOX$ el mejor resultado es para la búsqueda exhaustiva. Un $RBOX > 1$ indica que los límites del frente de Pareto han sido sobrepasados, fenómeno que suele ocurrir cuando la convergencia del algoritmo no ha sido buena como se puede observarse en la figura 4.8 para el caso de los frentes obtenidos por los algoritmos de búsqueda aleatoria y exhaustiva.

  Adicionalmente y con el objetivo de comparar los resultados producidos por los algoritmos ϵ-MOGA y ϵ-MOEA, con mayor profundidad, se determinan los indicadores alcance comparativo $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$[20] y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$[21] entre las 10 optimizaciones llevadas a cabo. La tabla 4.3 muestra los resultados, donde se puede ver que ϵ-MOGA supera claramente a ϵ-MOEA[22].

[20] Indica la proporción, sobre uno, de individuos de ϵ-MOGA que son dominados por individuos de ϵ-MOEA.

[21] Indica la proporción, sobre uno, de individuos de ϵ-MOEA que son dominados por individuos de ϵ-MOGA.

[22] Vence el que mayor valor de alcance comparativo presenta.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	HR	SP	RBOX
ϵ-MOGA						
mínimo	42	41	0.00098609	0.98126	7.3169e-7	0.92096
máximo	42	42	0.0010467	0.98132	1.1942e-6	0.92417
media	**42**	**41.9**	**0.0010133**	**0.98129**	**9.6254e-7**	0.92237
std	0	0.31623	2.1838e-5	1.4907e-5	1.5423e-7	0.0010894
ϵ-MOEA						
mínimo	37	37	0.00097226	0.9742	7.4842e-7	0.72001
máximo	42	42	0.0012437	0.98134	1.8873e-5	0.94469
media	40.7	39.7	0.0010765	0.97986	4.6762e-6	0.88361
std	1.567	1.767	8.0832e-005	0.002201	6.0982e-6	0.068235
B. Aleatoria						
mínimo	20	1	0.024967	1.0123	0.00057296	0.97196
máximo	28	3	0.046414	1.1193	0.001087	1.1163
máximo	24.5	2	0.036573	1.0544	0.00080837	1.0604
std	2.273	0.8165	0.00756	0.035804	0.0001666	0.045199
B. Exhaustiva						
mínimo	13	1	0.0258	0.91815	0.0038046	1.0349
máximo	16	3	0.029261	0.93981	0.0056666	1.0561
media	14.5	2	0.027531	0.92898	0.0047356	**1.0455**
std	2.1213	1.4142	0.0024474	0.015318	0.0013167	0.014933

Tabla 4.2: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$, IND, DG, HR, SP y RBOX resultado de la optimización del problema MOP2 con ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$
mínimo	0	0
máximo	0.02381	0.19512
media	0.0047619	**0.046875**
std	0.010039	0.063489

Tabla 4.3: Valores mínimos, máximos, medios y desviación típica de los indicadores comparativos $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ sobre las 10 optimizaciones llevadas a cabo con los algoritmos ϵ-MOGA y ϵ-MOEA, sobre el problema MOP2.

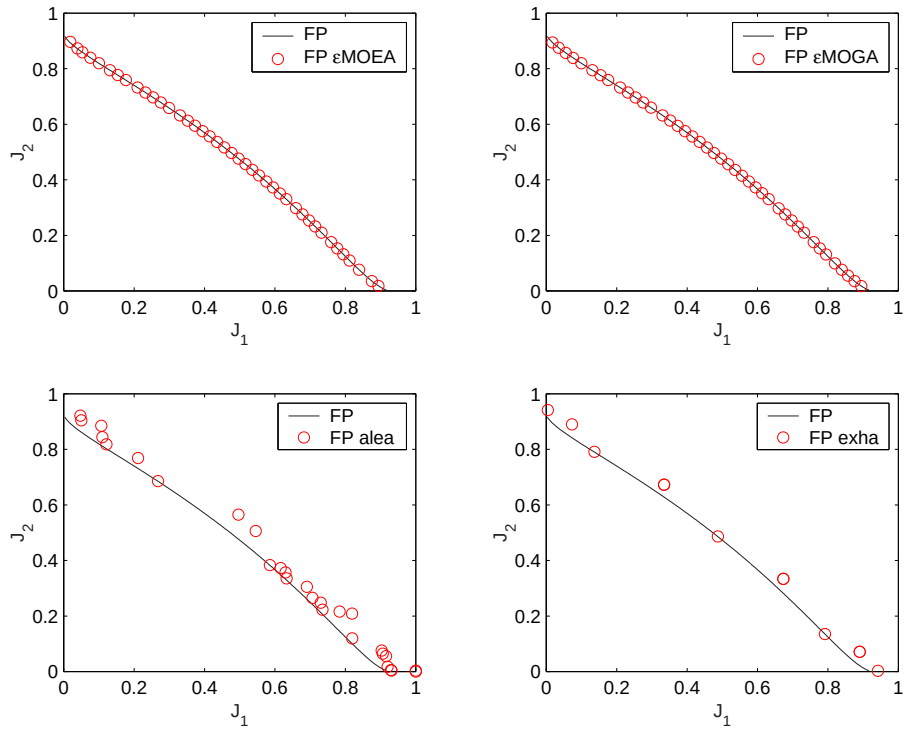

Figura 4.8: Frentes de Pareto obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva para MOP2.

- **Problema MOP3**. Para este problema, en concreto, se utilizan los siguientes parámetros para los algoritmos ϵ-MOGA y ϵ-MOEA:

  - $J_1^{max} = 18$ y $J_2^{max} = 25$.
  - $\epsilon_1 = 0.07$ y $\epsilon_2 = 0.07$.
  - $t_{max} = 375$ para el algoritmo ϵ-MOGA y $t_{max} = 1500$ para el algoritmo ϵ-MOEA de manera que se evalúen 1600 veces las funciones de test. El número de evaluaciones es bastante menor que en los problemas MOP1 y MOP2 porque el espacio de soluciones en MOP2 es más reducido y sólo de dimensión $\mathcal{R}^2$.

  Ahora $N = 1600$ para la búsqueda aleatoria y dado que espacio de soluciones es de dimensión $\mathcal{R}^2$ se escoge, para el algoritmo de búsqueda exhaustiva, tres valores para el parámetro q, $q = \{39, 40, 41\}$ por lo tanto $N = q^L = \{1521, 1600, 1681\}$.

  La tabla 4.4 muestra los resultados de la optimización del problema MOP3 con los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, respectivamente, obteniéndose las siguientes conclusiones:

  - Los algoritmos de búsqueda exhaustiva y aleatoria presentan resultados aceptables, debido a que el espacio de búsqueda no es muy grande. Sin embargo, los algoritmos ϵ-MOGA y ϵ-MOEA mejoran las prestaciones de los anteriores, en

número de individuos pertenecientes al frente (indicador IND), en convergencia sobre el frente (indicador DG) y diversidad (indicador SP) no siendo así con el indicador RBOX, es decir, la búsqueda exhaustiva y la aleatoria cubren mejor los extremos del frente.

- Comparando ϵ-MOGA y ϵ-MOEA se podría decir que los resultados obtenidos en número de individuos encontrados sobre el frente, en los parámetros HR y DG, que indican convergencia, y en diversidad son muy próximos aunque ϵ-MOGA es ligeramente superior a ϵ-MOEA.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	HR	SP	RBOX
ϵ-MOGA						
mínimo	50	29	0.004371	0.92598	0.043189	0.80731
máximo	55	47	0.059088	1.0087	0.19491	0.903743
media	**53.4**	**39.8**	**0.015815**	**0.96054**	**0.063213**	0.83798
std	1.4298	4.8028	0.01753	0.025081	0.047523	0.032019
ϵ-MOEA						
mínimo	42	29	0.004797	0.94033	0.022057	0.76353
máximo	55	45	0.15455	1.0312	0.17456	0.93093
media	49.6	38.8	0.022263	0.96032	0.065871	0.83743
std	3.9777	5.0509	0.046553	0.026563	0.043754	0.043954
B. Aleatoria						
mínimo	23	0	0.14994	0.91358	0.16575	1.0319
máximo	29	5	0.59095	1.1304	0.72441	1.2941
media	26.5	2.4	0.29669	1.0376	0.37274	1.1715
std	2.3688	1.4298	0.13664	0.067165	0.17527	0.087334
B. Exhaustiva						
mínimo	23	4	0.10635	0.97179	0.11693	1.0559
máximo	30	14	0.20709	1.0532	0.27289	1.0961
media	26.667	10.333	0.14936	1.0075	0.18033	**1.0783**
std	3.5119	5.5076	0.051953	0.041577	0.081967	0.020501

Tabla 4.4: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$, IND, DG, HR, SP y RBOX resultado de la optimización del problema MOP3 con ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$
mínimo	0	0
máximo	0.26415	0.17391
media	**0.091683**	0.043792
std	0.082066	0.05209

Tabla 4.5: Valores mínimos, máximos, medios y desviación típica de los indicadores comparativos $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ sobre las 10 optimizaciones llevadas a cabo con los algoritmos ϵ-MOGA y ϵ-MOEA, sobre el problema MOP3.

La figura 4.9 muestra los frentes obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

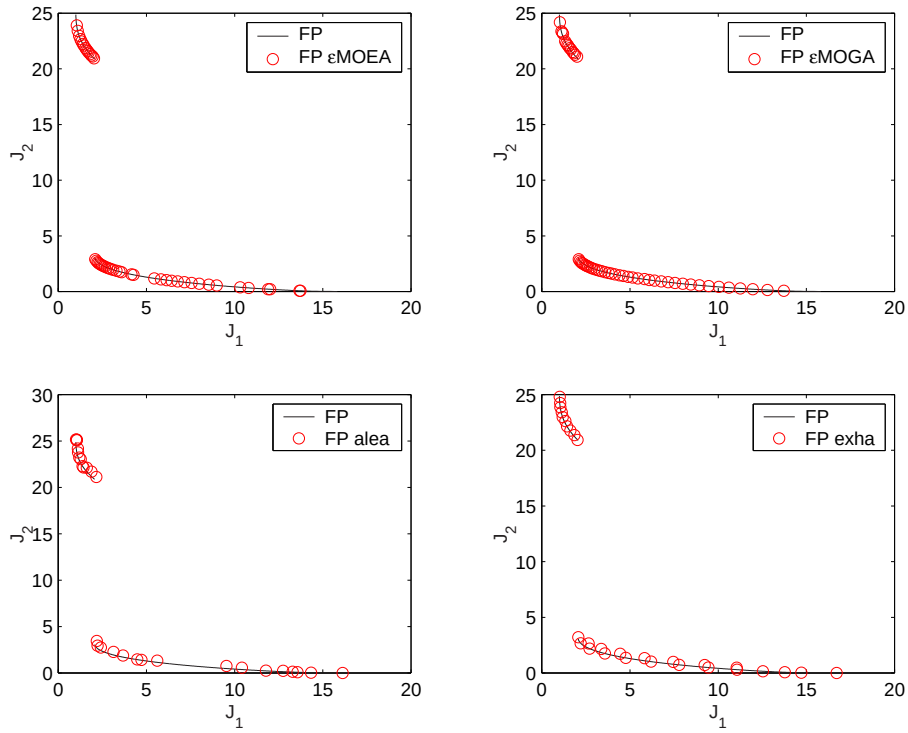

Figura 4.9: Arriba frentes de Pareto obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, (debajo) frentes de Pareto para búsqueda aleatoria y exhaustiva, para MOP3.

Los indicadores $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ entre las 10 optimizaciones llevadas a cabo se muestran en la tabla 4.5, donde se puede ver que ϵ-MOEA supera ligeramente a ϵ-MOGA.

- **Problema MOP4**. Para este problema, en concreto, se utilizan los siguientes parámetros para los algoritmos ϵ-MOGA y ϵ-MOEA:

  - $J_1^{max} = 7$ y $J_2^{max} = 12$.
  - $\epsilon_1 = 0.07$ y $\epsilon_2 = 0.075$.
  - $t_{max} = 4975$ para el algoritmo ϵ-MOGA y $t_{max} = 19900$ para el algoritmo ϵ-MOEA de manera que se evalúen 20000 veces las funciones de test.

  Como en MOP3, $N = 1600$ para la búsqueda aleatoria y para el algoritmo de búsqueda exhaustiva $q = \{39, 40, 41\}$ por lo tanto $N = q^L = \{1521, 1600, 1681\}$.

  La tabla 4.6 muestra los resultados de la optimización del problema MOP4 con los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, respectivamente. Del análisis de los resultados se pueden obtener las siguientes conclusiones:

  - Los algoritmos ϵ-MOGA y ϵ-MOEA son superiores a los algoritmos de búsqueda exhaustiva y aleatoria como en el caso anterior, excepto para el parámetro

RBOX que indica que los algoritmos de búsqueda exhaustiva y aleatoria capturan mejor los extremos de frente.

- De nuevo los resultados obtenidos por los algoritmos ϵ-MOGA y ϵ-MOEA son similares aunque si se observa con detalle se puede ver que ϵ-MOGA vence a ϵ-MOEA en todos los indicadores.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	HR	SP	RBOX
ϵ-MOGA						
mínimo	48	45	0.0025828	0.89691	0.010779	0.48617
máximo	72	67	0.0033632	0.99227	0.015627	1.0004
media	**69**	**53**	**0.0029958**	**0.98036**	**0.011898**	0.93883
std	7.4087	6.6667	0.00026354	0.029354	0.0014919	0.1591
ϵ-MOEA						
mínimo	47	41	0.0023925	0.89439	0.010879	0.48519
máximo	72	56	0.0037403	0.99122	0.034262	0.99565
media	67.3	49.7	0.0030903	0.97508	0.016875	0.93237
std	7.6891	5.2504	0.0003705	0.029501	0.0079281	0.15758
B. Aleatoria						
mínimo	22	0	0.27176	1.0379	0.011918	0.93984
máximo	34	0	0.35471	1.0923	0.10071	0.97942
media	26.7	0	0.31778	1.0615	0.054338	0.95699
std	3.6833	0	0.025936	0.016739	0.033288	0.014352
B. Exhaustiva						
mínimo	18	0	0.10642	0.94597	0.081473	0.93712
máximo	32	6	0.37318	1.0855	0.31214	1.0138
media	25	3	0.2398	1.0157	0.19681	**0.97546**
std	9.8995	4.2426	0.18863	0.098641	0.16311	0.054215

Tabla 4.6: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$, IND, DG, HR, SP y RBOX resultado de la optimización del problema MOP4 con ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$
mínimo	0.069444	0.041667
máximo	0.17143	0.085714
media	**0.12759**	0.056121
std	0.040821	0.016169

Tabla 4.7: Valores mínimos, máximos, medios y desviación típica de los indicadores comparativos $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ sobre las 10 optimizaciones llevadas a cabo con los algoritmos ϵ-MOGA y ϵ-MOEA, sobre el problema MOP4.

La figura 4.10 muestra los frentes obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva donde se puede observar la buena convergencia y diversidad obtenida por el algoritmo ϵ-MOGA en especial.

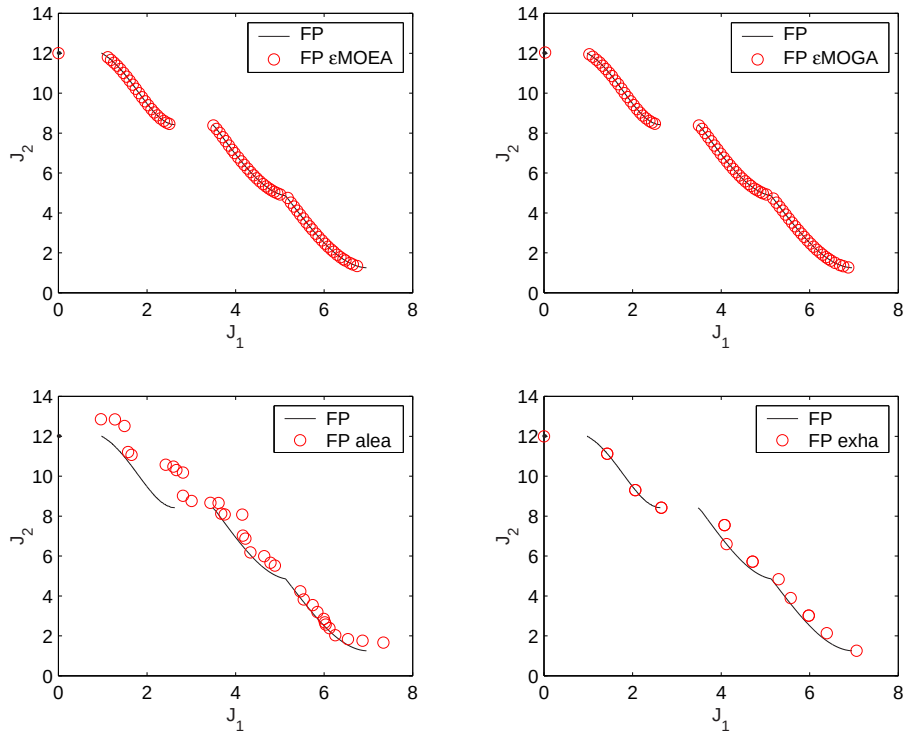

Figura 4.10: Frentes de Pareto obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva para MOP4.

Los indicadores $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ entre las 10 optimizaciones llevadas a cabo se muestran en la tabla 4.7, donde se puede ver que ϵ-MOEA supera ligeramente a ϵ-MOGA como ocurría en MOP3.

- **Problema MOP5**. Para este problema, en concreto, se utilizan los siguientes parámetros para los algoritmos ϵ-MOGA y ϵ-MOEA:

  - $J_1^{max} = 10$, $J_2^{max} = 2.5$ y $J_3^{max} = 0.5$.
  - $\epsilon_1 = 0.07$, $\epsilon_2 = 0.005$ y $\epsilon_2 = 0.005$.
  - $t_{max} = 375$ para el algoritmo ϵ-MOGA y $t_{max} = 1500$ para el algoritmo ϵ-MOEA de manera que se evalúen 1600 veces las funciones de test, igual que en MOP3, ya que el espacio de soluciones de MOP5 es reducido y de dimensión $\mathcal{R}^2$.

El espacio de soluciones es de dimensión $\mathcal{R}^2$ así que se escoge, para el algoritmo de búsqueda exhaustiva, dos valores para el parámetro q, $q = 141$ por lo tanto $141^3 = 19683$ y $q = 28$ por lo tanto $28^3 = 21952$.

La tabla 4.8 muestra los resultados de la optimización del problema MOP5 con los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, respectivamente.

Del análisis de los resultados se pueden obtener las siguientes conclusiones:

- Al igual que ocurría con MOP3, los algoritmos búsqueda exhaustiva y aleatoria presentan resultados aceptables, debido a que el espacio de búsqueda no es muy grande. Sin embargo, los algoritmos ϵ-MOGA y ϵ-MOEA mejoran las prestaciones de los anteriores, en número de individuos pertenecientes al frente (indicador IND) y en convergencia sobre el frente (indicador DG). No es así para el indicador RBOX, en el que, la búsqueda exhaustiva y la aleatoria ganan, ya que cubren mejor los extremos del frente ni tampoco para el indicador SP.
- Comparando ϵ-MOGA y ϵ-MOEA se observa que los resultados obtenidos con ϵ-MOGA son superiores en casi todos los indicadores.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	SP	RBOX
ϵ-MOGA					
mínimo	51	42	0.0013126	0.00067313	0.10373
máximo	75	69	0.0053024	0.069244	0.73901
media	**67.5**	**53.6**	**0.0036498**	0.018217	0.60578
std	8.436	8.5531	0.0014218	0.020046	0.2537
ϵ-MOEA					
mínimo	50	22	0.0031401	0.00075881	0.10683
máximo	64	38	0.0086726	0.11636	0.8018
media	57.9	30.6	0.0053153	0.028189	0.64123
std	4.3576	4.9933	0.0016498	0.033829	0.19646
B. Aleatoria					
mínimo	43	16	0.025463	0.0073439	0.84318
máximo	68	29	0.044492	0.027218	1.3446
media	55.1	20.9	0.032562	**0.017008**	1.2077
std	7.325	4.4083	0.0065214	0.0056923	0.14383
B. Exhaustiva					
mínimo	39	19	0.019014	0.013331	0.95409
máximo	50	22	0.023944	0.024009	1.181
media	44	20	0.021673	0.017204	**1.105**
std	5.5678	1.7321	0.0024877	0.0059115	0.13072

Tabla 4.8: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$, IND, DG, HR, SP y RBOX resultado de la optimización del problema MOP5 con ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva.

La figura 4.11 muestra los frentes obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva, donde de nuevo se puede observar la buena convergencia y distribución del frente obtenido por el algoritmo ϵ-MOGA.

De nuevo, para comparar los resultados producidos por los algoritmos ϵ-MOGA y ϵ-MOEA, se determinan los indicadores comparativos $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ entre las 10 optimizaciones llevadas a cabo. La tabla 4.9 muestra los resultados, donde se puede apreciar, en este caso, que el algoritmo ϵ-MOGA supera claramente a ϵ-MOEA.

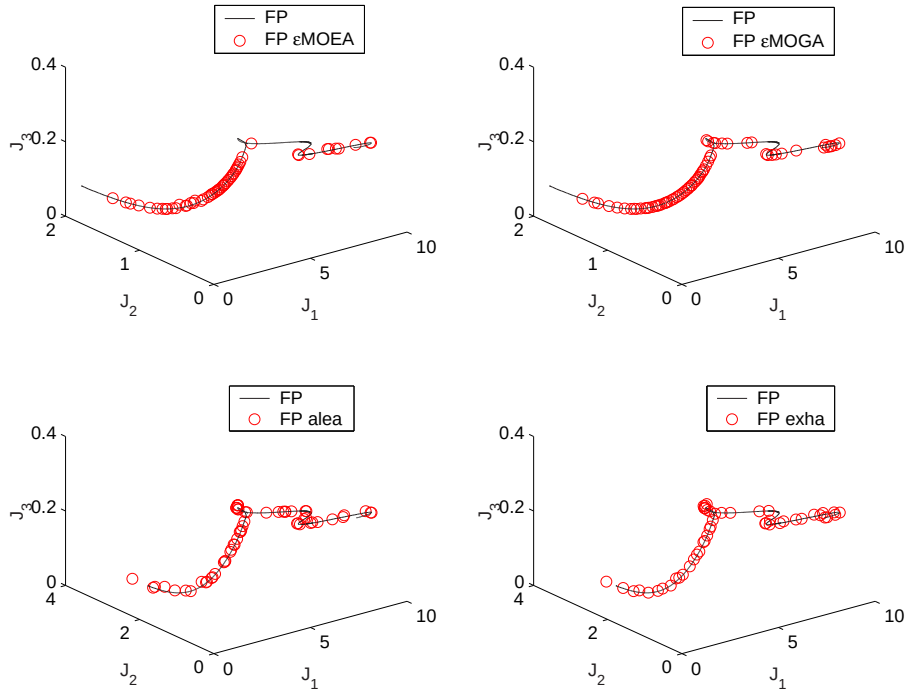

Figura 4.11: Frentes de Pareto obtenidos por los algoritmos ϵ-MOGA, ϵ-MOEA, búsqueda aleatoria y exhaustiva para MOP5.

	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$	$\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$
mínimo	0	0.083333
máximo	0.088235	0.24074
media	0.041026	**0.1719**
std	0.033116	0.055813

Tabla 4.9: Valores mínimos, máximos, medios y desviación típica de los indicadores comparativos $\mathcal{C}(\Theta_\epsilon^{\epsilon MOEA}, \Theta_\epsilon^{\epsilon MOGA})$ y $\mathcal{C}(\Theta_\epsilon^{\epsilon MOGA}, \Theta_\epsilon^{\epsilon MOEA})$ sobre las 10 optimizaciones llevadas a cabo con los algoritmos ϵ-MOGA y ϵ-MOEA, sobre el problema MOP5.

Una vez vistos y analizados los resultados obtenidos por los algoritmos anteriores se presentan las siguientes conclusiones:

- Los algoritmos ϵ-MOGA y ϵ-MOEA presentan mejores prestaciones que los algoritmos de búsqueda aleatoria y exhaustiva para todos los problemas.
- Los algoritmos de búsqueda aleatoria y exhaustiva sólo presentan resultados aceptables en problemas donde el espacio de soluciones es reducido (p.e. los problemas MOP3 y MOP5).
- El algoritmo ϵ-MOGA presenta mejores valores que ϵ-MOEA prácticamente en todos los indicadores ($|\Theta_{P_\epsilon}|$, IND, DG, SP y $RBOX$) de los MOP1...MOP5. En cuanto a los indicadores de alcance comparativo ϵ-MOEA es superior a ϵ-MOGA en MOP1 y MOP4 y ϵ-MOGA sobre ϵ-MOEA en MOP2 y MOP5.
- El principal inconveniente de los algoritmos ϵ-MOGA y ϵ-MOEA es que no caracterizan adecuadamente los extremos del los frentes de Pareto. Este hecho es el que ha motivado el desarrollo del algoritmo $\not\epsilon$-MOGA que será descrito en la siguiente sección.

4.5. Espacio del fenotipo variable. $\not\epsilon$-MOGA

En la sección anterior, se pusieron de manifiesto los buenos resultados obtenidos por el algoritmo ϵ-MOGA sobre los problemas MOP1...MOP5 en cuanto a convergencia y diversidad sobre el frente de Pareto se refiere. Sin embargo, el algoritmo ϵ-MOGA (al igual que ϵ-MOEA) presenta una serie de inconvenientes:

- Para determinar la cota máxima de $|\Theta_{P_\epsilon}|$ es necesario conocer J_i^{max}, es decir, los límites superiores de las funciones de *fitness*. Es fácil determinar dichos valores en funciones de tipo test, donde $\mathbf{J}(\Theta_P)$ es conocido. Sin embargo, en problemas reales no siempre es posible y se suele terminar sobredimensionando dicho valor.
- El algoritmo presenta problemas a la hora de caracterizar los extremos del frente de Pareto, como consecuencia implícita de la aplicación del concepto ϵ-dominante. Por ejemplo, si se observa con detalle la solución obtenida por ϵ-MOGA para MOP1 en su extremo (ver figura 4.12) se observa que la solución $\mathbf{J}(\theta)$ *box* domina todos las posibles soluciones que pudieran ser localizadas en la zona de color gris. De ahí que se impida la correcta caracterización de los extremos del frente. Aunque se podría mejorar reduciendo el valor de ϵ, se generaría un mayor número de *boxes* y un aumento de la memoria y el coste computacional necesarios.

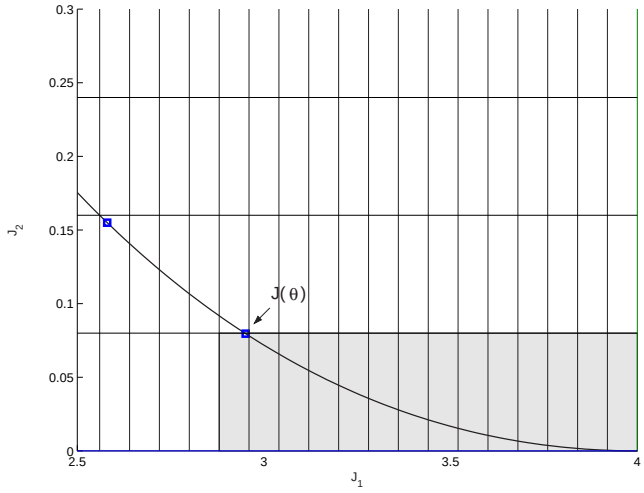

Figura 4.12: Detalle de uno de los extremos del frente de Pareto para MOP1.

Con el objetivo de paliar estas deficiencias del algoritmo ϵ-MOGA, se modifica éste para que:

- Los límites del frente de Pareto sean dinámicos. Si los límites son dinámicos y se quiere seguir manteniendo una cota máxima de la memoria necesaria para almacenar Θ_{P_ϵ}, es necesario fijar el número de *boxes* en cada dimensión, por lo tanto, ϵ ahora será variable y se ajustará en función de los límites dinámicos del frente $J_i^{min} \leq J_i(\Theta_{P_\epsilon}) \leq J_i^{max}$ y del número de divisiones propuestas para cada dimensión. Siendo $J_i^{min} = \text{mín}\, J_i(\Theta_{P_\epsilon})$ y $J_i^{max} = \text{máx}\, J_i(\Theta_{P_\epsilon})$.

- Asegurar que los extremos no sean eliminados por el efecto de ϵ-dominante.

Para ello, es necesario realizar las siguientes modificaciones sobre algunas definiciones y sobre el algoritmo ϵ-MOGA.

Ahora no será necesario fijar los diferentes J_i^{max} y los correspondientes ϵ_i. A cambio, habrá que fijar el número de *boxes* que se desean para cada función de *fitness*, es decir, los n_box_i. Esto hace que la definición de *box* se vea modificada quedando ahora de la siguiente manera.

Definición 4.8 (Box para ϵ-MOGA): *Dado un vector θ en el espacio de soluciones, cuya imagen en el espacio de las funciones de fitness es $\mathbf{J}(\theta)$, con un número de divisiones n_box_i para cada dimensión i, se define el vector $\mathbf{box}(\theta) = [box_1(\theta) \ldots box_s(\theta)]$ donde:*

$$box_i(\theta) = \left\lceil \frac{J_i(\theta) - J_i^{min}}{J_i^{max} - J_i^{min}} \cdot n\_box_i \right\rceil \ \forall i \in [1 \ldots s]$$

y $J_i^{min} \leq J_i(\theta) \leq J_i^{max}$.

$\triangle$

Esto hace que $box_i(\theta) \in [0 \dots n\_box_i]$ y que ϵ_i (el ancho de los *boxes*) sea:

$$\epsilon_i = \frac{J_i^{max} - J_i^{min}}{n\_box_i}.$$

Por lo tanto, la ecuación (4.6) ahora quedaría de la siguiente manera:

$$|\Theta_{P_\epsilon}| \leq \frac{\prod_{i=1}^{s} n\_box_i + 1}{n\_box_{max} + 1}. \tag{4.8}$$

Gracias a la nueva definición de *box*, a las soluciones que determinen los extremos inferiores (para cada una de las funciones $J_i(\theta)$) del frente de Pareto se les asignará un valor de $box_i(\theta) = 0$, pues $J_i(\theta) = J_i^{min}$. De esta manera, ninguna solución θ^*, cuyo $J_i(\theta^*) > J_i^{min}$ podrá ϵ-dominarlas, ya que su $box_i(\theta^*) \geq 1$ por aplicación de la nueva definición de *box*. Por lo tanto, se resuelve uno de los problemas del algoritmo ϵ-MOGA.

Adicionalmente, el procedimiento de inclusión de un individuo θ en el archivo $A(t)$[23] tiene que ser ampliado, dado que los límites J_i^{min} y J_i^{max} tienen que ir adaptándose.

Definición 4.9 (Inclusión de θ en $A(t)$ para ϵ-MOGA): *Dado un vector de soluciones θ y el archivo $A(t)$, θ será incluido en el archivo si y sólo si θ sería incluido por el procedimiento de inclusión de la definición 4.7 cuando se aplica de la siguiente manera:*

1. *Constituir el conjunto $A'(t)$ que contendrá los individuos no dominados del conjunto $A(t) \bigcup \theta$.*
2. *Vaciar $A(t)$.*
3. *Determinar los nuevos límites de manera que $J_i^{min} = \min A'(t)$ y $J_i^{max} = \max A'(t)$ $\forall w$.*
4. *Aplicar el procedimiento de inclusión de la definición 4.7 para intentar introducir, uno por uno, los individuos de $A'(t)$ en $A(t)$.*

$\triangle$

El procedimiento general de inclusión anterior puede ser aplicado de forma más eficiente para cubrir las siguientes situaciones (ver figura 4.13):

Caso A: $J_i^{min} < J_i(\theta) \leq J_i^{max}$ $\forall i \in w := [1 \dots s]$. Se resuelve aplicando directamente el procedimiento de inclusión descrito en la definición 4.7. Este caso incluya o no al individuo θ no modifica los límites J_i^{min} y J_i^{max}.

Caso B: $J_i(\theta) > J_i^{max}$ $\forall i \in w$. El individuo θ es dominado por al menos un individuo en el archivo, por lo tanto el individuo directamente no es incluido. Este caso, como es obvio, no modifica los límites J_i^{min} y J_i^{max}.

[23] $A(t)$ almacena Θ_{P_ϵ}. En el peor de los casos $A(t)$ contendrá al menos una solución, que definirá los límites del espacio del frente de Pareto.

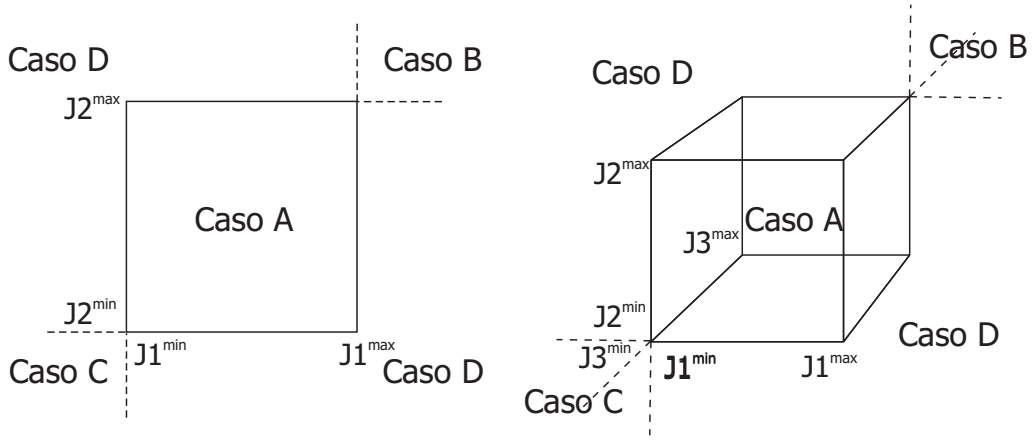

Figura 4.13: Diferentes casos que se pueden plantean al intentar incluir un individuo $\mathbf{J}(\theta)$ en el archivo $A(t)$. A la izquierda caso bidimensional, a la derecha tridimensional.

Caso C: $J_i(\theta) < J_i^{min}\ \forall i \in w$. El individuo θ domina a todos los individuos que pudiesen estar incluidos en el archivo, por lo tanto, hay que proceder de la siguiente manera:

1. Vaciar $A(t)$.
2. Incluir el individuo θ en el archivo $\Rightarrow A(t) = \theta$.
3. Establecer los nuevos límites de manera que $J_i^{min} = J_i^{max} = J_i(\theta)\ \ \forall i \in w$.

Caso D: Se da cuando no se cumple ninguno de los casos anteriores. Hay que proceder como en el caso general de la definición 4.9.

Por lo tanto, el algoritmo $\not\epsilon^{\nearrow}$-MOGA, resultará de la aplicación de las nuevas definiciones de *box* e inclusión anteriores que afectan única y exclusivamente a la función *guardar* que se ejecuta en las líneas de código 5 y 9 del algoritmo ϵ-MOGA.

Para analizar el comportamiento del algoritmo $\not\epsilon^{\nearrow}$-MOGA se optimizan las funciones MOP1 a MOP5 utilizando los mismos parámetros que se utilizaron con ϵ-MOGA. La diferencia es que ahora no es necesario establecer ni los límites J_i^{max} para cada función, ni el valor de ϵ_i, a cambio hay que establecer el parámetro n_box_i. Por lo tanto, se establecen los n_box_i [24] de la siguiente manera:

- MOP1. $n\_box_1 = n\_box_2 = 50$.
- MOP2. $n\_box_1 = n\_box_2 = 50$.
- MOP3. $n\_box_1 = 257$ y $n\_box_2 = 357$.
- MOP4. $n\_box_1 = 100$ y $n\_box_2 = 160$.

[24] Han sido elegidos para que los ϵ_i resultantes sean equivalentes a los establecidos anteriormente para ϵ-MOGA.

- MOP5. $n\_box_1 = 100$, $n\_box_2 = 500$ y $n\_box_3 = 100$.

La tabla 4.10 muestra los resultados obtenidos de la optimización de los problemas MOP1 a MOP5 con el algoritmo $\not\nearrow$-MOGA (valores mínimos, máximos, medios y desviación típica sobre 10 optimizaciones de cada problema). En negrita aparecen aquellos indicadores medios que son mejores que los obtenidos con el algoritmo ϵ-MOGA y si lo son también sobre ϵ-MOEA, búsqueda exhaustiva y aleatoria aparecen además subrayados. Del análisis de los resultados de la tabla se pueden extraer las siguientes conclusiones:

- El algoritmo $\not\nearrow$-MOGA obtiene valores de $RBOX$ muy próximos a la unidad con una buena convergencia (parámetros IND, DG y HR), lo que pone de manifiesto que los extremos de los frentes han sido bien capturados. Para algunos problemas (p.e. MOP2 o MOP3) $RBOX > 1$ debido a que $\not\nearrow$-MOGA ha capturado mejor los extremos que incluso el frente real[25].
- El indicador HR también ha mejorado como consecuencia de que los extremos han sido bien capturados.
- En cuanto al resto de parámetros, ha empeorado levemente el indicador SP de diversidad, en gran medida debido a que al capturar los extremos la distribución de las soluciones a lo largo del frente no es tan uniforme.

Las figuras 4.14, 4.15 y 4.16 muestran los frentes obtenidos por el algoritmo $\not\nearrow$-MOGA, donde se puede observar la buena convergencia y distribución de los frentes obtenidos, especialmente en sus extremos.

El algoritmo $\not\nearrow$-MOGA ha demostrado:

- Ser más flexible que ϵ-MOGA, ya que no es necesario establecer los límites J_i^{max}, pues se obtienen de forma dinámica
- Buena convergencia y diversidad.

Sin embargo, que los límites del frente se modifiquen dinámicamente, hace que no sea posible demostrar que el resultado que se obtiene sea un conjunto ϵ-Pareto, es decir, no se puede asegurar que $A(t) \in \mathcal{P}_\epsilon(\Theta)$. Mediante la figura 4.17 se pretende poner de manifiesto una situación que sirva para aclarar este tema.

En la parte superior izquierda (subfigura (a)) se muestra una situación en concreto donde $A(t)$ contiene 5 soluciones (representadas mediante puntos) y se plantea la inclusión, utilizando la definición 4.9, de una nueva solución θ^1 (representada mediante un cuadrado). θ^1 cumple las condiciones para ser incluida en el archivo (dado que θ^1 es una solución no dominada) y modificar los límites del frente (caso D). Por lo tanto, se modifican los límites J_2^{min} y J_1^{max}, se crea el nuevo *grid* y se incluyen los individuos de $A(t) \bigcup \theta^1$ que

[25]Hay que tener en cuenta que el frente real se ha obtenido utilizando una búsqueda exhaustiva con un *grid* de anchura muy pequeña, es decir, que no es el frente real sino una muy buena aproximación a éste.

	$\lvert\Theta_{P_\epsilon}\rvert$	IND	DG	HR	SP	RBOX
MOP1						
mínimo	27	25	0.0034297	0.92962	0.059411	0.99931
máximo	27	27	0.0040955	0.92983	0.060563	1.001
media	**27**	**26.1**	0.0038679	**0.92976**	0.059702	**1.0001**
std	0	0.8756	0.00023498	6.8921e-005	0.00028116	0.0005185
MOP2						
mínimo	45	21	0.0019596	0.98249	5.7279e-6	1.0371
máximo	47	44	0.0029061	0.98658	7.3761e-5	1.0979
media	**46.4**	37.7	0.002307	**0.98362**	1.6977e-5	**1.048**
std	0.69921	7.5579	0.00033038	0.0012752	2.0352e-5	0.019114
MOP3						
mínimo	50	35	0.007825	0.9022	0.07253	0.94274
máximo	61	47	0.099646	1.052	0.5724	1.2391
media	**53.6**	**40.6**	0.053144	**0.97751**	0.30873	**1.11**
std	3.6271	3.4059	0.033257	0.046255	0.16899	0.095079
MOP4						
mínimo	66	12	0.0038673	0.98857	0.0093337	0.98676
máximo	74	36	0.009598	0.99396	0.014055	1.0026
media	**71.5**	24.6	0.0063108	**0.99074**	**0.011659**	**0.9973**
std	2.1213	7.1988	0.0019695	0.0015925	0.0016385	0.0054074
MOP5						
mínimo	97	73	0.0041362		0.0010988	0.21786
máximo	126	100	0.0083422		0.019644	1.1851
media	**103.8**	**82.6**	0.0065137		**0.010923**	**0.98925**
std	8.728	8.9839	0.0015376		0.0056474	0.30764

Tabla 4.10: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_{P_\epsilon}\rvert$, IND, DG, HR, SP y RBOX resultado de la optimización de los problemas MOP1 a MOP5 con ϵ-MOGA.

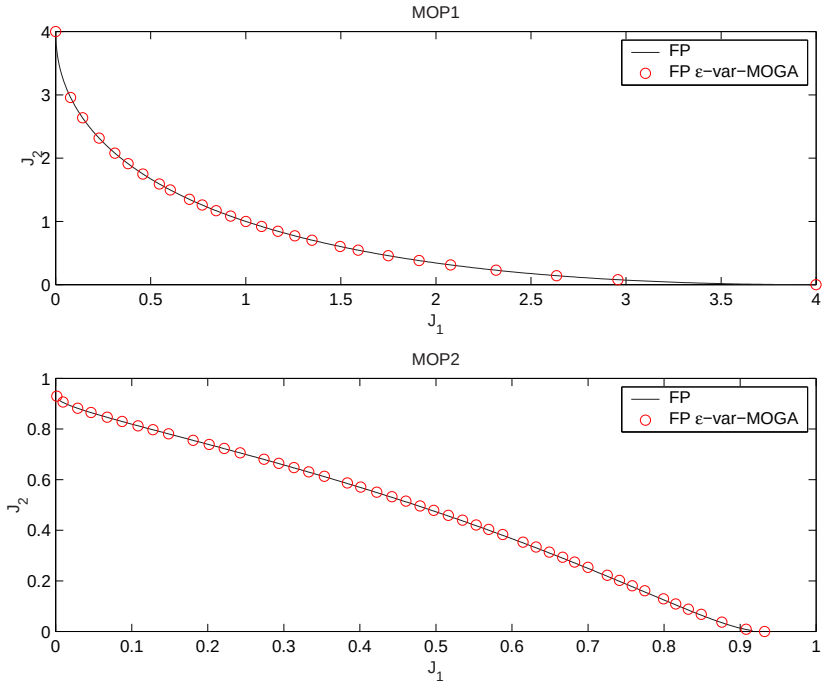

Figura 4.14: Frentes de Pareto obtenidos por el algoritmo ε-MOGA para MOP1 y MOP2.

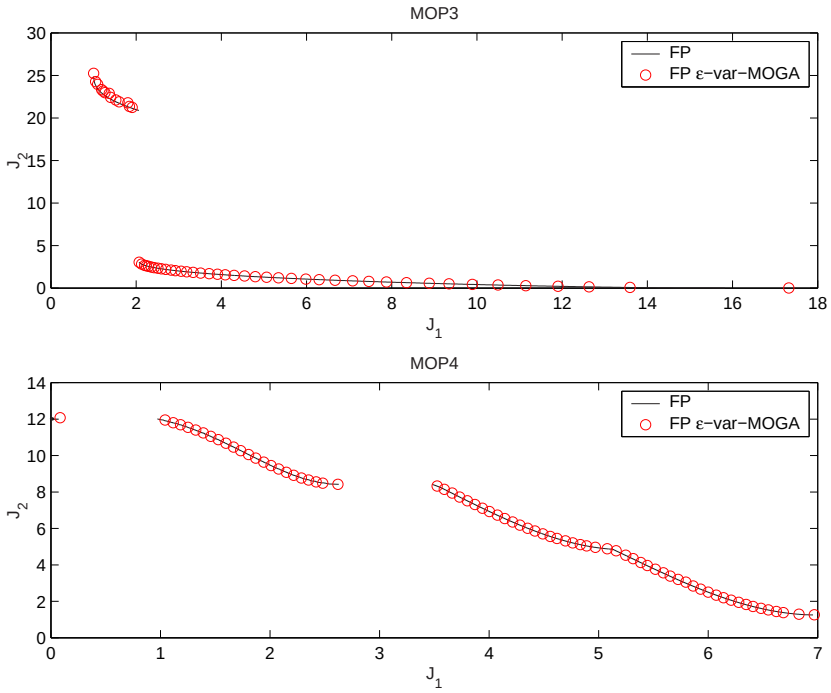

Figura 4.15: Frentes de Pareto obtenidos por el algoritmo ε-MOGA para MOP3 y MOP4.

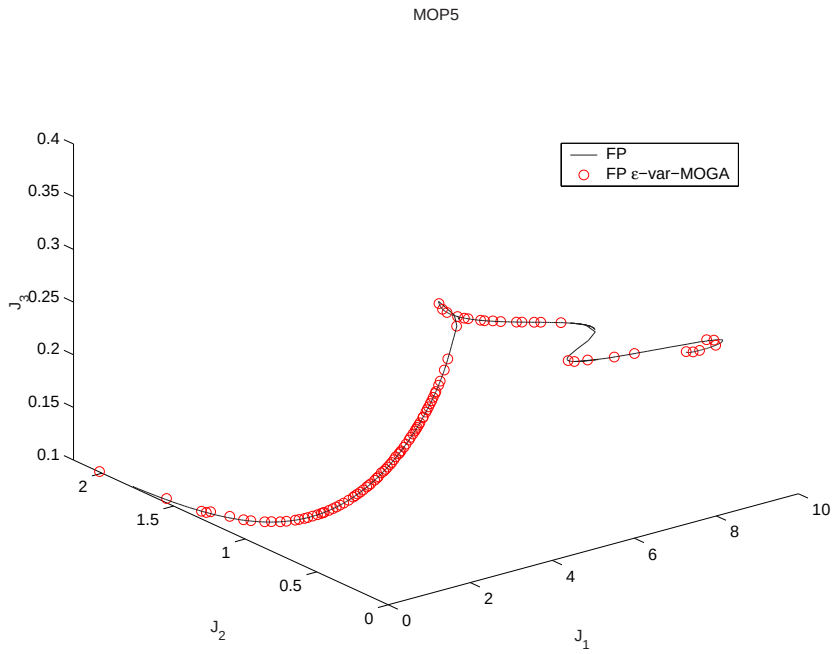

Figura 4.16: Frentes de Pareto obtenidos por el algoritmo ε-MOGA para MOP5.

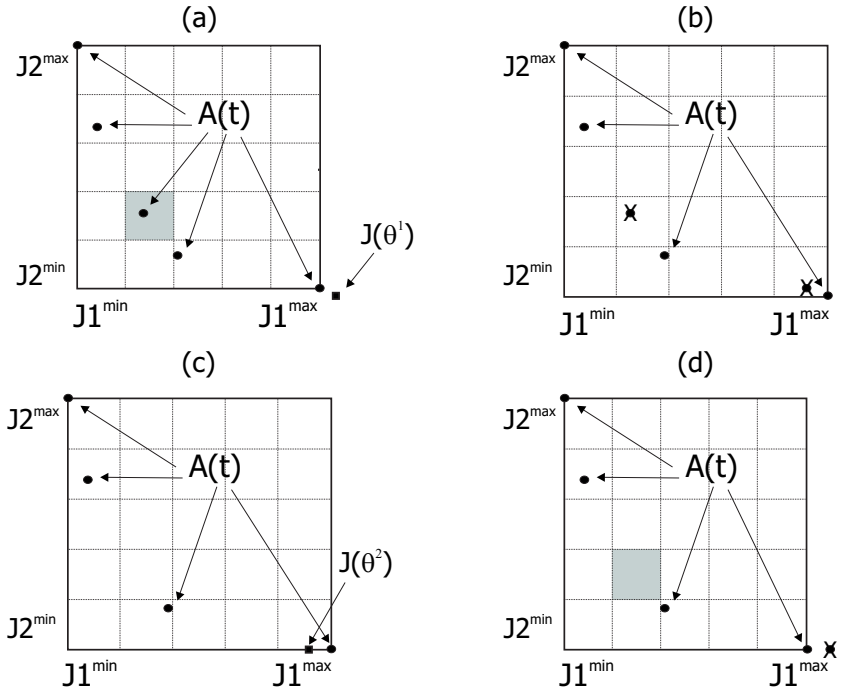

Figura 4.17: Ilustración del fenómeno de deterioro producido como consecuencia de la modificación de los límites J_2^{min} y J_1^{max} debido a la inclusión de θ^1 (en primer instancia) y de θ^2 (posteriormente) en $A(t)$.

proceda, es decir, los no ϵ-dominados. El resultado se muestra en la parte superior derecha (subfigura (b)) donde se han representado mediante puntos tachados las soluciones que son eliminadas de $A(t)$ porque resultan ser ϵ-dominadas.

A continuación, se plantea la inclusión de otra nueva solución θ^2 (representada mediante un cuadrado en la subfigura (c)) repitiéndose la misma situación que antes, es decir, θ^2 cumple las condiciones para ser incluida (θ^2 es no dominada) y modificar de nuevo los límites del frente (caso D). El resultado de aplicar el procedimiento de inclusión se muestra en la subfigura (d). Ahora el archivo $A(t)$ contiene 4 soluciones, se ha sombreado un *box* que ha quedado vacío (el *box* [2,2]) y que podría estar ocupado por una solución (la que se encuentra dentro del *box* sombreado de la subfigura (a)) que se ha perdido porque en una situación previa resulto ϵ-dominada, produciéndose así un deterioro del frente resultante.

Esto puede justificar, en parte, que el parámetro SP crezca, ya que se pueden perder soluciones internas del frente, debido a que al modificar los límites J_i^{max}, J_i^{min}, puede que soluciones que son ϵ-dominadas en un determinado momento no lo fuesen posteriormente, pero sin embargo se pierden y por lo tanto, si se da esta situación el frente resultante $A(t)$ no sería un conjunto ϵ-Pareto.

Para solucionar este problema se plantean dos alternativas[26]:

- La primera alternativa pasa por utilizar los algoritmos ϵ-MOGA y ↗-MOGA secuencialmente. Primero se utiliza el algoritmo ↗MOGA que determina los límites del frente y después una vez haya terminado se utilizan los límites del frente como parámetros del algoritmo ϵ-MOGA que al ser ejecutado aseguraría no deterioro y que $A(t) \in \mathcal{P}_\epsilon(\Theta)$. Evidentemente, las soluciones de los extremos conseguidas por el algoritmo ↗-MOGA son almacenadas ya que ϵ-MOGA no las alcanzaría.

- La segunda alternativa es más sofisticada y utiliza los algoritmos ϵ-MOGA y ↗-MOGA al mismo tiempo. El procedimiento sería el siguiente:

  - En una fase inicial se ejecuta el algoritmo ↗MOGA (que irá adaptando los límites del frente) vigilando que no se pierdan soluciones no dominadas que en un determinado momento resultasen ϵ-dominadas. El propio algoritmo ↗MOGA lo asegura, ya que, todas las soluciones no dominadas terminan almacenándose en la población $P(t)$ (aunque sean eliminadas del archivo $A(t)$) por el propio procedimiento aplicado por la función *actualizar* (línea 10 del algoritmo). Llegará un momento en que $P(t)$ sólo contenga soluciones no dominadas y se desee introducir otra solución no dominada. En ese momento, deja de ejecutarse el algoritmo ↗-MOGA y se inicia la segunda fase.
  - En la segunda fase, se ejecuta el algoritmo ϵ-MOGA (utilizando las poblaciones $P(t)$ y el archivo $A(t)$ que dejó el algoritmo ↗-MOGA así como los límites del

[26]Estas alternativas serían utilizadas si se desease asegurar que $A(t) \in \mathcal{P}_\epsilon(\Theta)$. Que decir que el algoritmo ↗-MOGA no lo puede demostrar aunque sí que ha mostrado resultados aceptables de forma empírica a la hora de resolver los problemas MOP1...MOP5.

frente) a partir de la línea 5 de código, es decir, se analiza que individuos en $P(t)$ pueden, ahora, formar parte de $A(t)$. De esta manera, se asegura que ningún individuo no dominado, que pudiese formar parte del archivo, por no ser ϵ-dominado por ninguno de sus miembros, se perdiese. A partir de ese momento, el algoritmo ϵ-MOGA es el que se encargará (sin modificar los límites del frente) de obtener el archivo definitivo que ahora sí será un conjunto ϵ-Pareto.

La primera alternativa conlleva un coste computacional mayor que la segunda, pues es necesario ejecutar una vez el algoritmo ϵ-MOGA y otra el algoritmo $\not\epsilon$-MOGA, pero tendrá las ventajas de ambos algoritmos. Asegurará que $A(t) \in \mathcal{P}_\epsilon(\Theta)$, que se caracterizen correctamente los extremos del frente y que no sea necesario establecer unos límites máximos para el mismo.

La segunda alternativa presenta, un menor coste computacional y, aunque asegura que la solución $A(t) \in \mathcal{P}_\epsilon(\Theta)$, puede que la población $P(t)$ se llene demasiado pronto de soluciones no dominadas y no se llegue a caracterizar correctamente los límites del frente. Para reducir en la medida de lo posible este efecto se puede optar por aumentar el parámetro $Nind_P$ (reduciendo el número de iteraciones t_{max} para mantener el mismo coste computacional) y evitar que se repitan individuos en la población $P(t)$.

4.6. Ejemplo. Estructura de tres barras

El problema de optimización está relacionado con la estructura mecánica de tres barras de la figura 4.18 [117, 15]. Los parámetros de la estructura son L=1 m, $\beta = 45^o$, $\alpha = 30^o$ y $F = 20kN$. El objetivo es diseñar las secciones de las tres barras $\theta = [\theta_1, \theta_2, \theta_3]$ para minimizar, por una parte, el volumen total de las mismas $J_1(\theta)$ y una combinación lineal de los desplazamientos del nodo P, $J_2(\theta)$.

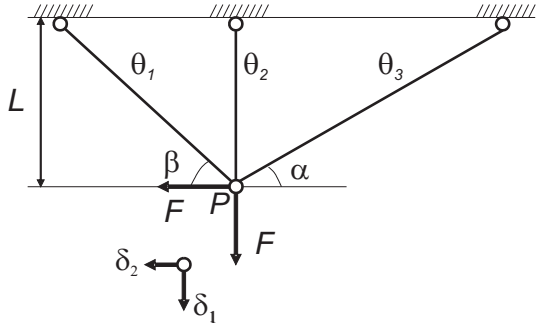

Figura 4.18: Estructura mecánica con tres barras $\beta = 45^o$ y $\alpha = 30^o$.

$$\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)] \tag{4.9}$$
$$0.1 \cdot 10^{-4} m^2 \leq \theta_i \leq 2 \cdot 10^{-4} m^2, i = 1 \ldots 3$$

donde:

$$J_1(\theta) = L(\frac{a_1}{\sin\beta} + a_2 + \frac{a_3}{\sin\alpha}), \tag{4.10}$$
$$J_2(\theta) = 0.25\delta_1 + 0.75\delta_2. \tag{4.11}$$

Las deformaciones δ_1 y δ_2 se pueden calcular de la siguiente manera:

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \frac{L}{E} \begin{bmatrix} \gamma_{11} & \gamma_{12} \\ -\gamma_{12} & \gamma_{22} \end{bmatrix}^{-1} \begin{bmatrix} F \\ -F \end{bmatrix}, \tag{4.12}$$

donde $E = 200GPa.$ es el módulo de *Young* y:

$$\begin{aligned} \gamma_{11} &= \theta_2 + \theta_1 \sin^3\beta + \theta_3 \sin^3\alpha, \\ \gamma_{12} &= \theta_1 \sin^2\beta\cos\beta - \theta_3 \sin^2\alpha\cos\alpha, \\ \gamma_{22} &= -\theta_1 \sin\beta\cos^2\beta - \theta_3 \sin\alpha\cos^2\alpha. \end{aligned}$$

Adicionalmente se establecen 3 restricciones relacionadas con las fuerzas de reacción de cada barra de manera que el máximo estrés permitido, para cada barra, es $\sigma = 200MPa.$

$$\frac{|N_i|}{\theta_i} \leq \sigma, i = 1 \ldots 3, \tag{4.13}$$
$$\tag{4.14}$$

donde N_i (la fuerza de reacción de cada barra) se calcula de la siguiente manera:

$$N_1 = \frac{a_1 E}{L}(\delta_1 \sin\beta - \delta_2 \cos\beta)\sin\beta, \tag{4.15}$$
$$N_2 = \frac{a_2 E}{L}\delta_1, \tag{4.16}$$
$$N_3 = \frac{a_3 E}{L}(\delta_1 \sin\alpha + \delta_2 \cos\alpha)\sin\alpha. \tag{4.17}$$

Para resolver el problema de optimización planteado en (4.9) se utilizará el algoritmo $\not{c}$MOGA con los siguientes parámetros.

- $Nind_G = 4$ y $Nind_P = 100$.
- $t_{max} = 4975$. De está manera el número de evaluaciones de la función objetivo $\mathbf{J}(\theta)$ será 20000.
- $P_{c/m} = 0.1$.
- $d_{ini} = 0.25$, $d_{fin} = 0.1$, $\beta_{ini} = 10.0$ y $\beta_{fin} = 0.1$.

- $n\_box_1 = n\_box_2 = 100$.

Las restricciones descritas en (4.15) se tratarán utilizando funciones de penalización estáticas (ver sección 2.10). De manera que, las funciones a optimizar quedarían de la siguientes manera:

$$J_1(\theta) = L(\frac{a_1}{\sin\beta} + a_2 + \frac{a_3}{\sin\alpha}) + \sum_{i=1}^{3} G_i(\theta), \tag{4.18}$$

$$J_2(\theta) = 0.75\delta_1 + 0.25\delta_2 + \sum_{i=1}^{3} G_i(\theta), \tag{4.19}$$

donde:

$$G_i(\theta) = max\left[0,\ \frac{|N_i|}{\theta_i} - \sigma\right]$$

La figura 4.19 muestra el resultado de la optimización donde se puede observar que el frente de Pareto, es cóncavo y discontinuo al igual que el conjunto de óptimos de Pareto.Si fuese necesaria una mayor caracterización del frente bastaría con aumentar el parámetro n_box_i[27].

Con el frente de Pareto ya disponible se podría escoger, de él, la solución θ^* que más cerca estuviese del vector ideal. El vector ideal, en este caso, es

$$\mathbf{J}^{ideal} = [0.067464cm., 285.73cm^2.]$$

y

$$\theta^* = \arg \min_{\theta \in \Theta_{P_\epsilon}} ||\mathbf{J}(\theta) - \mathbf{J}^{ideal}||_2,$$

que para este caso resultaría ser[28]

$$\theta^* = [0.1e^{-4}, 0.4633e^{-4}, 1.1263e^{-4}] \ \Rightarrow\ \mathbf{J}(\theta^*) = [0.15469cm., 285.731cm.^2].$$

[27]La modificación de n_box_i no supone tener que evaluar un número mayor de veces las funciones objetivo.

[28]Para calcular $||\mathbf{J}(\theta) - \mathbf{J}^{ideal}||_2$ se han escalado las magnitudes $J_1(\theta)$ y $J_2(\theta)$ en el rango $[0 \ldots 1]$.

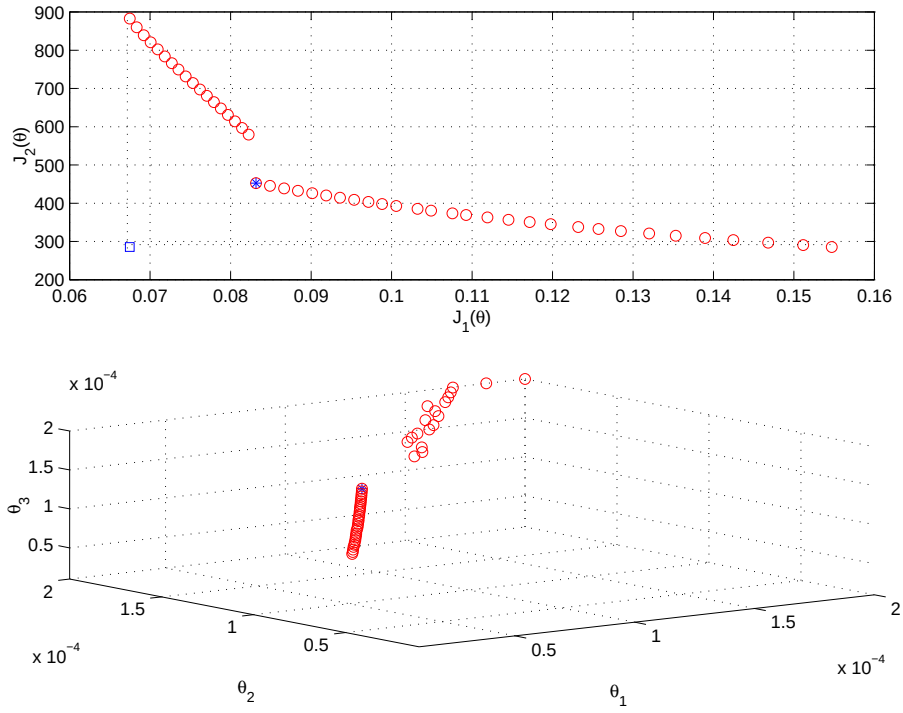

Figura 4.19: Arriba el frente de Pareto $\mathbf{J}(\Theta_{P_\epsilon})$ para el problema de las tres barras. $J_1(\theta)$ y $J_2(\theta)$ representados en cm y cm^2 respectivamente. $\square$ representa el vector ideal $\mathbf{J}^{ideal} = [0.067464, 285.73]$ y * es el punto de frente de Pareto más cercano al vector ideal. Debajo el conjunto de óptimos de Pareto Θ_{P_ϵ} que generada dicho frente.

4.7. Conclusiones

A lo largo de este capítulo se ha presentado un algoritmo genético multiobjetivo elitista llamado ϵ-MOGA que utiliza una arquitectura paralela maestro-esclavo y operadores de cruce y mutación con ajuste de parámetros basados en funciones. Se ha realizado un estudio comparativo estándar, a través de la optimización de los problemas MOP1 a MOP5 que, ha demostrado que ϵ-MOGA mejora en prestaciones a los algoritmos ϵ-MOEA, búsqueda aleatoria y exhaustiva.

El algoritmo ϵ-MOGA está basado en el concepto de ϵ-dominancia que le permite:

- Obtener, como solución al problema de optimización, un conjunto ϵ-Pareto que converge hacia el conjunto de óptimos de Pareto Θ_P. Ya que se ha demostrado que no produce deterioro.
- Utilizar recursos de memoria limitada.
- Una buena distribución de las soluciones a lo largo del frente de Pareto.

Se ha presentado una variante original del algoritmo ϵ-MOGA, el algoritmo $\not\epsilon\nearrow$MOGA que aunque no puede demostrar la convergencia de la solución de forma teórica, sí lo ha hecho de forma empírica con la optimización de los problemas MOP1 a MOP5 y el ejemplo de la estructura con tres barras. El algoritmo, al igual que ϵ-MOGA, mantiene recursos de memoria limitada y además:

- Ajustar dinámicamente los límites del frente de Pareto.
- Evitar que las soluciones que pertenecen a los extremos del frente se pierdan por la propia aplicación del concepto *box* dominante.

Adicionalmente, se han propuesto dos enfoques diferentes basados en el uso de los algoritmos ϵ-MOGA y $\not\epsilon\nearrow$-MOGA con el objetivo de aunar las características de ambos algoritmos ya puntualizas.

Capítulo 5

Algoritmo Evolutivo ϵ-GA

5.1. Introducción

Este capítulo tiene como principal objetivo presentar la implementación de un EA denominado ϵ-GA diseñado para optimizar funciones mono objetivo multimodales que presenten múltiples (incluso infinitos) óptimos globales. Evidentemente, si existen infinitos óptimos globales no será posible encontrarlos todos, pero sí una muestra de éstos que los caractericen adecuadamente.

Este problema, aunque diferente, guarda bastantes similitudes con el problema de optimización multiobjetivo abordado en el capítulo anterior, de ahí que el algoritmo ϵ-GA esté inspirado en el algoritmo ϵ-MOGA y en alguna de las características presentadas en la sección 2.11 (Optimización de Funciones Multimodales). Por lo tanto, los objetivos del algoritmo ϵ-GA serán:

1. Convergencia de la solución determinada hacia el conjunto de óptimos globales.
2. Que la solución determinada sea una muestra bien distribuida del conjunto de óptimos globales.
3. Uso de memoria limitada para que su implementación sea viable y los costes computacionales asequibles.

Para cumplir con este último objetivo, el algoritmo utilizará un *grid* de tamaño ϵ sobre el espacio de soluciones[1]. Por otra parte, guardando similitudes con el algoritmo ϵ-MOGA, ϵ-GA mantendrá su misma estructura basada en las tres poblaciones $P(t)$ (principal), $A(t)$ (archivo donde se mantendrá la solución al problema de optimización) y $G(t)$ (auxiliar).

El resto del capítulo se distribuye de la siguiente manera: la sección 5.2 describe la terminología que será utilizada en el resto del capítulo así como el procedimiento que gestionará el contenido del archivo $A(t)$, justificando los objetivos arriba planteados. En la sección 5.3 se presenta el algoritmo ϵ-GA desarrollado. La siguiente sección, sección 5.4 evalúa el algoritmo ϵ-GA utilizando un conjunto de funciones multimodales de test y compara los resultados con dos estrategias basadas en el algoritmo SQP.

5.2. Conceptos relacionados con ϵ-GA

Antes de describir el algoritmo $\epsilon - GA$ se procede a mostrar una serie de definiciones que serán de gran ayuda para su posterior presentación.

Para empezar, el problema de optimización a abordar sería el siguiente:

Definición 5.1 (Conjunto mínimo global): *Dado un dominio finito $D \subseteq \mathcal{R}^L$, $D \neq \emptyset$ y una función a optimizar (función fitness) $J : D \rightarrow \mathcal{R}$, el conjunto Θ^* será el conjunto mínimo global de J si y sólo si Θ^* contiene todos los mínimos globales de J.*

[1] A diferencia del algoritmo ϵ-MOGA donde el *grid* se utilizaba en el espacio de funciones.

$$\Theta^* := \{\theta \in D : J(\theta) = J^*\}, \tag{5.1}$$

siendo J^* un mínimo global de J para el espacio de búsqueda D.

△

De esta definición se deduce que Θ^* es un conjunto único. La determinación de Θ^* resulta imposible si se asume que, puede tratarse de un conjunto no finito e inconexo cuya determinación conllevaría, en el peor de los casos, la exploración de todo el espacio de búsqueda D. En este caso, las mejores expectativas pasan por determinar un conjunto finito Θ^*_ϵ que sea una aproximación discretizada de Θ^* sobre el espacio de soluciones D, que lo caracterice adecuadamente. Para ello, se divide el espacio de soluciones mediante un *grid* con *boxes* de anchura ϵ_i para cada una de las dimensiones $i \in [1 \dots L]$ y se obliga (al igual que ocurría con el algoritmo ϵ-MOGA) a que sólo pueda existir una solución en un mismo *box* del *grid*. De esta manera, gracias al *grid* se fuerza a que las soluciones en Θ^*_ϵ estén distribuidas y puedan caracterizar Θ^* como se planteaba en el segundo objetivo al principio del capítulo.

Por lo tanto, para dar con Θ^*_ϵ habrá que concretar los conceptos de aproximación y discretización, para ello se definirán a continuación los conceptos quasi mínimo global y *box* representante.

Definición 5.2 (Quasi mínimo global): *Dado un dominio finito $D \neq \emptyset$ y una función a optimizar $J : D \to \mathcal{R}$, la solución θ se considera quasi mínimo global de J si y sólo si*

$$J(\theta) \leq J^* + \delta, \tag{5.2}$$

siendo $\delta > 0$ y J^* un mínimo global de J.

△

De la definición anterior se deduce que una solución global mínima es también una solución global quasi mínima.

Definición 5.3 (Conjunto quasi mínimo global): *Dado un dominio finito $D \neq \emptyset$ y una función a optimizar $J : D \to \mathcal{R}$, el conjunto Θ^{**} será el conjunto quasi mínimo global de J si y sólo si Θ^{**} contiene todas las soluciones globales quasi mínimas de J.*

$$\Theta^{**} := \{\theta \in D : J(\theta) \leq J^* + \delta\}, \tag{5.3}$$

siendo J^* un mínimo global de J para el espacio de búsqueda D y $\delta > 0$.

△

De esta definición se deduce que $\Theta^* \subseteq \Theta^{**}$. Además si $\delta \to 0$ entonces $\Theta^{**} \to \Theta^*$.

Las nuevas definiciones de *box* y *box*-representante en el espacio de soluciones son las siguientes:

Definición 5.4 (Box para ϵ-GA): *Dado un vector θ en el espacio de soluciones $D \subseteq \mathcal{R}^L$, se define su box para $\epsilon_i > 0$ como el vector $\mathbf{box}(\theta) = [box_1(\theta) \dots box_L(\theta)]$ donde:*

$$box_i(\theta) = \left\lfloor \frac{\theta_i - \theta_i^{min}}{\epsilon_i} \right\rfloor \forall i \in [1 \dots L].$$

$\triangle$

Esto hace que $box_i(\theta) \in [0 \dots (n\_box_i - 1)]$, siendo n_box_i el número de divisiones del *grid* en la dimensión i

$$n\_box_i = \left\lceil \frac{\theta_i^{max} - \theta_i^{min}}{\epsilon_i} \right\rceil,$$

donde θ_i^{max} y θ_i^{min} determinan los límites del espacio de soluciones D y $(\theta_i^{max} - \theta_i^{min}) \geq \epsilon_i$.

Definición 5.5 (*box*-representante): *Dado dos vectores θ^1, θ^2 en el espacio de soluciones, cuyas imágenes en el espacio de la función de fitness son $J(\theta^1)$ y $J(\theta^2)$ respectivamente, se dirá que θ^1 box-representa a θ^2 (denotado como $\theta^1 \preceq_{\boxdot} \theta^2$) para un determinado $\epsilon_i > 0$ si*

$$\mathbf{box}(\theta^1) = \mathbf{box}(\theta^2) \wedge J(\theta^1) \leq J(\theta^2). \tag{5.4}$$

$\triangle$

Y por lo tanto, se puede proceder a definir Θ_ϵ^* de la siguiente manera:

Definición 5.6 (Conjunto mínimo ϵ-global): *Dado un conjunto de soluciones Θ en el espacio de soluciones, el conjunto $\Theta_\epsilon^* \subseteq \Theta$ será un conjunto mínimo ϵ-global de Θ si y sólo si*

1. *Contiene únicamente soluciones globales quasi mínimas de Θ*

$$\Theta_\epsilon^* \subseteq (\Theta \cap \Theta^{**}).$$

2. *Cualquier vector en $\Theta \cap \Theta^{**}$ tiene un box-representante en Θ_ϵ^*, es decir:*

$$\forall \theta \in \Theta \cap \Theta^{**}, \;\; \exists \theta^* \in \Theta_\epsilon^* : \theta^* \preceq_{\boxdot} \theta.$$

$\triangle$

De la definición anterior se deduce que, dado un conjunto Θ, Θ_ϵ^* no tiene porque ser un conjunto único. De hecho, esto podrá ocurrir cuando varias soluciones globales mínimas de Θ compartan el mismo box, ya que, serán mutuamente *box*-representantes una de la otra.

Definición 5.7 (Conjunto $\Phi_\epsilon(\Theta)$): *Al conjunto de todos los conjuntos mínimos ϵ-globales de Θ se le denominará como $\Phi_\epsilon(\Theta)$.*

$\triangle$

Con estas definiciones es posible establecer el procedimiento que gestionará el contenido del archivo $A(t)$ (donde se almacenará la solución Θ_ϵ^* del problema de optimización[2]). Para ello es necesario conocer, por una parte, los parámetros ϵ_i y δ (que serían parámetros del diseñador[3]) y adicionalmente también el mínimo global J^* lo cual no es siempre posible. La mejor aproximación a J^* que el algoritmo puede determinar será J_Θ^{min}, es decir aquella que menor valor de la función J ha producido.

$$J_\Theta^{min} = \min_{\theta \in \Theta} J(\theta). \tag{5.5}$$

Definición 5.8 (Inclusión de θ en $A(t)$ para ϵ-GA): *Dado un vector en el espacio de soluciones θ y el archivo $A(t)$, θ será incluido en el archivo si y sólo si*

$$J(\theta) \leq J_{A(t)}^{min} + \delta \tag{5.6}$$

$$\wedge$$

$$\nexists \theta^* \in A(t) : \theta^* \preceq_\square \theta. \tag{5.7}$$

Al mismo tiempo, si θ es incluido en el archivo podría modificar $J_{A(t)}^{min}$ (mejor solución sometida al proceso de inclusión en $A(t)$), por lo tanto, serán eliminados de $A(t)$ todas las soluciones θ^* que cumplen la siguiente condición

$$J(\theta^*) > J_{A(t)}^{min} + \delta \tag{5.8}$$

$$\vee$$

$$\theta \preceq_\square \theta^*. \tag{5.9}$$

$\triangle$

Utilizando este procedimiento de actualización es posible establecer los siguientes teoremas y sus demostraciones:

Teorema 5.1: *Para cualquier conjunto Θ en el espacio de soluciones, que representa el conjunto de todos los individuos creados durante el proceso de optimización y cuyo espacio de soluciones se encuentra dividido por un grid con anchuras $\epsilon_i > 0$, es posible dar con un conjunto mínimo ϵ-global, con tamaño $|\Theta_\epsilon^*|$ acotado*

$$|\Theta_\epsilon^*| \leq \prod_{i=1}^{L} n\_box_i. \tag{5.10}$$

[2]Al igual que se hizo en el capítulo anterior con el algoritmo ϵ-MOGA.

[3]ϵ_i representa la anchura del box para la dimensión i y δ la cota establecida para considerar una solución como *quasi* mínima global.

Demostración. *Las ecuaciones (5.7) y (5.9) se encargan de asegurar que sólo pueda existir una solución en un mismo box (del grid) y dado que* $\prod_{i=1}^{L} n\_box_i$ *es el número total de boxes el teorema queda demostrado.*

$\Diamond$

La determinación de los coeficientes ϵ_i será dependiente del problema en cuestión y condicionará el tamaño máximo de individuos en el Θ_ϵ^* cumpliéndose así el tercero de los objetivos que se pretendía alcanzar con el algoritmo. Evidentemente, el diseñador debería ajustar dichos coeficientes en función del significado físico de las variables que componen el espacio de búsqueda.

Por otra parte, el procedimiento de inclusión de la definición 5.8 asegura, como posteriormente se demostrará, que el contenido del archivo $A(t)$ converge hacia un conjunto mínimo ϵ-global (el primero de los objetivos que se perseguía) siempre y cuando $J_{A(t)}^{min}$ converja hacia el mínimo global J^*. Para que esto último ocurra, como ya se comentó en la introducción del capítulo anterior, el algoritmo debe poder generar cualquier solución dentro del espacio de búsqueda con una probabilidad mayor que cero (lo cual se cumple) y ser elitista, es decir, la mejor solución no se debe perder, condición que también cumple el procedimiento anterior como se demuestra a continuación.

Teorema 5.2: *Para cualquier conjunto* Θ *en el espacio de soluciones, el contenido del archivo* $A(t)$ *resultante de la inclusión de los individuos de* Θ *sobre él, mediante el procedimiento descrito en la definición 5.8, mantendrá, al menos, una de las mejores soluciones de* Θ *asegurando así que el algoritmo es elitista*

$$\forall \Theta \in D, A(t) \cap \bar{\Theta} \neq \emptyset,$$

siendo

$$\bar{\Theta} := \{\bar{\theta} \in \Theta : J(\bar{\theta}) = J_\Theta^{min}\}$$

y por lo tanto $J_{A(t)}^{min} = J_\Theta^{min}$.

Demostración. *Para demostrar que el algoritmo es elitista hay que probar que* $A(t) \cap \bar{\Theta} \neq \emptyset$. *Para ello, se han de cumplir las siguientes condiciones:*

1. *Un individuo* $\theta^* \in A(t) \cap \bar{\Theta}$ *nunca debe ser eliminado por un individuo* $\theta \notin \bar{\Theta}$ *que se sometiese al procedimiento dado en la definición 5.8.*

2. *Si* $A(t) \cap \bar{\Theta} = \emptyset$ *y un individuo* $\theta \in \Theta \cap \bar{\Theta}$ *se somete al procedimiento de la definición 5.8 debe ser incluido en el mismo.*

La ***condición 1*** *implica, ya que* $\theta^* \in A(t) \cap \bar{\Theta}$, *que* $J_{A(t)}^{min} = J(\theta^*) = J_\Theta^{min}$ *y será cierta si no se cumple ninguna de las ecuaciones de eliminación (5.8) y (5.9). La ecuación (5.8) es imposible de cumplir para* $\delta > 0$, *ya que* $J_{A(t)}^{min} = J(\theta^*)$ *y la ecuación (5.9) sólo se cumpliría si* $\theta \in \bar{\Theta}$ *en cuyo caso se seguiría cumpliendo que* $A(t) \cap \bar{\Theta} \neq \emptyset$.

*La **condición 2** implica que* $J(\theta) = J_{\Theta}^{min} < J_{A(t)}^{min}$*, ya que* $A(t) \cap \bar{\Theta} = \emptyset \Rightarrow J_{A(t)}^{min} > J_{\Theta}^{min}$ *y* $\theta \in \Theta \cap \bar{\Theta} \Rightarrow J(\theta) = J_{\Theta}^{min}$*, por lo tanto, será cierta ya que las ecuaciones de inserción (5.6) y (5.7) se cumplen ambas para cualquier valor de* $\delta > 0$*.*

$\Diamond$

Si Θ contiene alguna solución mínima global, es decir $\Theta \cap \Theta^* \neq \emptyset$, entonces $J_{A(t)}^{min} = J_{\Theta}^{min} = J^*$. Se trata de un caso particular del planteado en el teorema 5.2, ya que si, $\Theta \cap \Theta^* \neq \emptyset$ entonces $\bar{\Theta} = \Theta \cap \Theta^*$ por la propia definición de mínimo global.

Teorema 5.3: *Para cualquier conjunto* Θ *en el espacio de soluciones, dividido por un grid con anchuras* $\epsilon_i > 0$ *y* $\Theta \cap \Theta^* \neq \emptyset$*, el contenido del archivo* $A(t)$ *resultante de la inclusión de los individuos de* Θ *sobre él, mediante el procedimiento descrito en la definición 5.8, será un conjunto mínimo* ϵ*-global*

$$A(t) \in \Phi_{\epsilon}(\Theta).$$

Demostración. *Se asume inicialmente que* $A(t) \notin \Phi_{\epsilon}(\Theta)$*. Para ello, según la definición 5.6, se debe cumplir cualquiera de las dos siguientes condiciones:*

1. *Existe un individuo en* $A(t)$ *que no es solución global quasi mínima, es decir,*

$$\exists \theta \in A(t) : J(\theta) > J^* + \delta.$$

2. *Existe una solución global quasi mínima en* Θ *que no tiene un box-representante en* Θ_{ϵ}^**, es decir,*

$$\exists \theta \in \Theta \cap \Theta^{**} : \nexists \theta' \in A(t) \mid \theta' \preceq_{\boxdot} \theta.$$

*La **condición 1** no se cumple ya que, la ecuación (5.6) asegura que ninguna solución que no sea global quasi mínima pueda ser insertada en el archivo una vez* $J_{A(t)}^{min} = J^*$*, situación que se dará cuando una solución global mínima contenida en* Θ *haya sido insertada en el archivo. Cuando esto ocurra la ecuación (5.8) asegura que todas las soluciones que no sean global quasi mínima sean eliminadas del archivo.*

*La **condición 2** tampoco se cumple, pues las ecuaciones (5.6) y (5.7) aseguran que una solución global quasi mínima* θ^{**} *es insertada a menos que tenga un box-representante en* $A(t)$*. Además esa solución box-representante (*θ^**) nunca será eliminada del archivo a menos que otra solución* θ' *la box-represente (ecuación (5.9)) en cuyo caso* θ' *box-representaría a ambas* θ^{**} *y* θ^**.*

$$\theta^* \preceq_{\boxdot} \theta^{**}, \theta' \preceq_{\boxdot} \theta^* \Rightarrow \theta' \preceq_{\boxdot} \theta^{**}.$$

Por lo tanto, asumir que $A(t) \notin \Phi_{\epsilon}(\Theta)$ *resulta falso.*

$\Diamond$

Para terminar con esta sección se analizará el efecto de los parámetros ϵ_i y δ dado el importante papel que éstos juegan en la determinación de Θ^*_ϵ, especialmente en el número de soluciones globales quasi óptimas que contendrá ($|\Theta^*_\epsilon|$).

Los coeficientes ϵ_i determinan el grado de discretización que se desea aplicar sobre Θ^*_ϵ y estarán relacionados directamente con el sentido físico de los parámetros que definen las diferentes dimensiones del espacio de búsqueda. A medida que $\epsilon_i \downarrow\downarrow$ aumentará $|\Theta^*_\epsilon|$. La figura 5.1 ilustra este fenómeno para la siguiente función de *fitness*[4]:

$$J(\theta) = \begin{cases} (\theta - 0.5)^2 - 0.09, & (\theta - 0.5)^2 - 0.09 > 0 \\ 0, & (\theta - 0.5)^2 - 0.09 \leq 0, \end{cases} \tag{5.11}$$

$$\theta \in D = [0 \ldots 1].$$

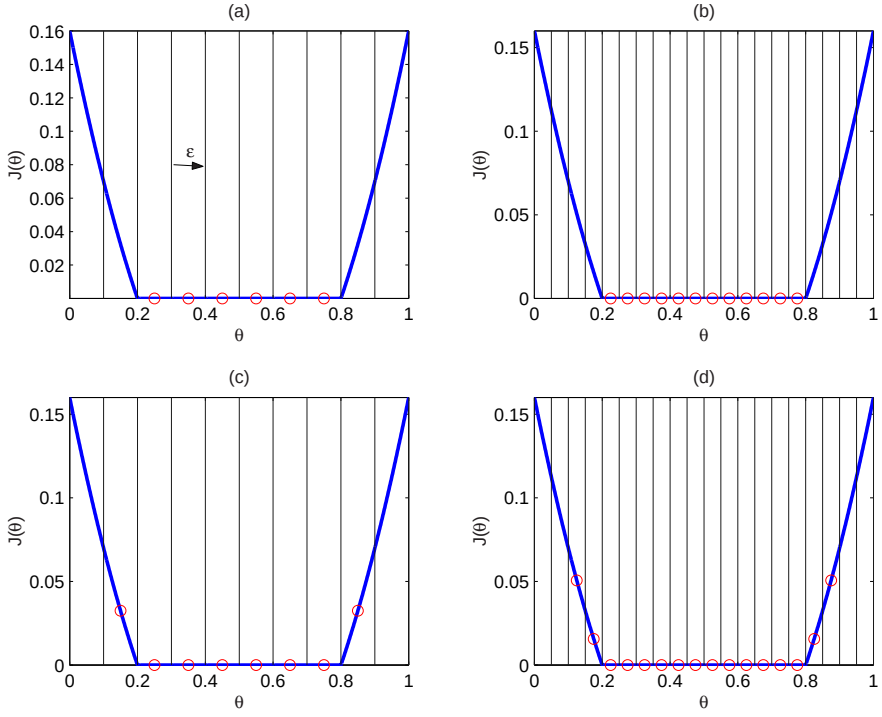

Figura 5.1: Ilustración de un posible contenido de Θ^*_ϵ (círculos) al optimizar la función 5.11. (a) $\delta = 0$ y $\epsilon = 0.1$. (b) $\delta = 0$ y $\epsilon = 0.05$. (c) $\delta = 0.06$ y $\epsilon = 0.1$. (d) $\delta = 0.06$ y $\epsilon = 0.05$.

Las subfiguras (a) y (b) ilustran la situación donde $\delta = 0$ por lo tanto $\Theta^* = \Theta^{**}$. En el caso (a) $\epsilon = 0.1$ y $|\Theta^*_\epsilon| = 6$, como se puede observar[5]. Mientras que, si se utiliza un $\epsilon = 0.05$, $|\Theta^*_\epsilon| = 12$ como muestra la subfigura (b)[6]. Si se analizan los dos mismos casos

[4]Se trata de una función muy sencilla unidimensional sin mínimos locales pero con infinitos óptimos globales y donde $\Theta^* := \{\theta \in D : \theta \in [0.2 \ldots 0.8]\}$.

[5]Sería el máximo número de soluciones que podría contener Θ^*_ϵ teniendo en cuenta el *grid* de la figura (las líneas verticales representan los límites de los *boxes* del *grid*).

[6]En este caso particular la relación entre ϵ y Θ^*_ϵ es lineal, aunque no se puede generalizar.

anteriores pero utilizando, por ejemplo, un $\delta = 0.06$ un posible resultado sería el ilustrado en las subfiguras (c) y (d). Ahora $\Theta^* \subset \Theta^{**}$ y $|\Theta^*_\epsilon| = 8$ cuando $\epsilon = 0.1$ y $|\Theta^*_\epsilon| = 16$ cuando $\epsilon = 0.05$. Por lo tanto, se puede ver que ambos parámetros δ y ϵ afectarán directamente sobre el número de soluciones contenidas en Θ^*_ϵ.

Lo ideal, como ya se comentó, es que $\Theta^*_\epsilon \Rightarrow \Theta^*$ y para ello $\delta \Rightarrow 0$, de ahí que δ ejerza un papel directamente relacionado con la convergencia. Sin embargo, el parámetro δ cumple otra función, ya que fomenta la diversidad en el contenido del archivo y por lo tanto, la capacidad de éste de caracterizar Θ^*, de ahí que, para fomentar este segundo aspecto no interese utilizar valores excesivamente pequeños. La figura 5.2 muestra dos situaciones que ponen de manifiesto que el valor del parámetro δ juega un papel importante fomentando diversidad en el archivo $A(t)$ en el proceso de convergencia hacía Θ^*. Para el ejemplo, $A(t)$ sólo contiene una solución (representada mediante $\circ$) y se analiza la inclusión, mediante el procedimiento de la definición 5.8, de otra solución (representada mediante $\square$).

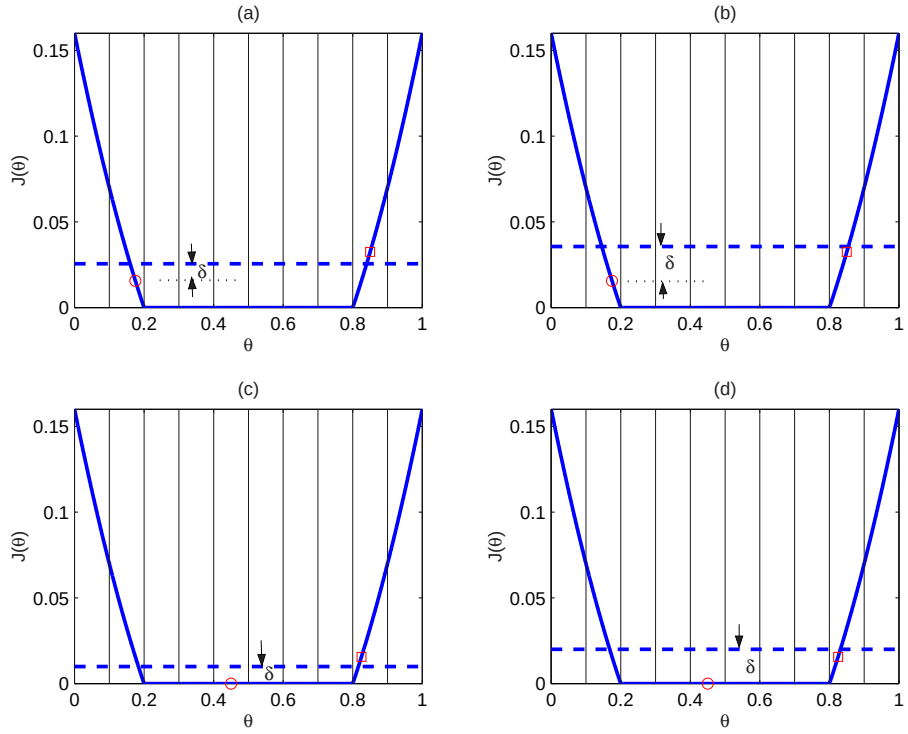

Figura 5.2: Escenificación de la inclusión de $\square$ en el archivo cuando $\delta = 0.01$ (casos (a) y (c)) y $\delta = 0.02$ (casos (b) y (d)). $\circ$ representa el contenido del archivo en cada situación.

En los casos (a) y (b) el archivo contiene una solución que no es solución global mínima, es decir, $J^{min}_{A(t)} \neq J^*$. Para el caso (a) $\delta = 0.01$ y $\delta = 0.02$ para el caso (b) (la línea horizontal a trazos representa el valor máximo que debe tener una solución θ para cumplir la ecuación (5.6)) lo que provoca que la solución representada mediante $\square$ pudiese ser insertada en el archivo en el caso (b) y no en el caso (a). Los casos (c) y (d) son equivalentes a los casos (a) y (b) respectivamente pero ahora el archivo contiene una solución global mínima $J^{min}_{A(t)} = J^{min}_{\Theta} = J^*$.

Por lo tanto, los casos (b) y (d) incluirían la solución $\square$ en el archivo y favorecería la caracterización de Θ^*, ya que la solución $\square$ favorece la exploración y caracterización de las soluciones mínimas globales entorno a $\theta = 0.8$, mientras que en los casos (a) y (c) no sería así.

Las conclusiones serían las siguientes:

- Un valor de $\delta \simeq 0$ fomentaría la convergencia y $\Theta_\epsilon^* \Rightarrow \Theta^*$ pero entorpecería la aproximación de Θ^* pudiendo empeorar su caracterización.
- Un valor de δ demasiado elevado tendría el efecto contrario ya que Θ_ϵ^*, aunque contendría soluciones globales quasi mínimas, podrían ser soluciones que más que caracterizar Θ^*, lo falseasen.

De ahí que, un buen planteamiento para escoger δ sea partir de un valor $\delta = \delta_{ini}$ e ir modificado δ (por ejemplo utilizando una función exponencial decreciente como la utilizada para el ajuste de algunos parámetros en el algoritmo ϵ-MOGA) hasta un valor $\delta = \delta_{fin}$ lo suficientemente pequeño como para que las soluciones globales quasi mínimas que se obtengan estén próximas a las soluciones globales mínimas.

Dado que δ irá siempre decreciendo, las soluciones que en un momento determinado estén en el archivo (por considerarse soluciones globales quasi mínimas) y desaparezcan al verse reducido δ (por dejar de ser soluciones globales quasi mínimas) nunca podrían considerarse soluciones globales quasi mínimas en un futuro por lo tanto las propiedades del procedimiento de inclusión 5.8 no se verán alteradas y $A(t) \in \Phi(\Theta)$.

5.3. Descripción de ϵ-GA

En este apartado se va a presentar el algoritmo ϵ-GA con el objetivo de dar con un conjunto mínimo ϵ-global, Θ_ϵ^*.

Las características del algoritmo son las siguientes:

- Se trata de un algoritmo elitista.
- No generacional, solapable y preservativo.
- Utiliza los mismos operadores de cruce y mutación que ϵ-MOGA.
- La selección se resuelve aleatoriamente y la determinación mediante el procedimiento de inclusión 5.8 y estado uniforme.
- La implementación utiliza una arquitectura paralela maestro-esclavo.
- Incorpora un mecanismo adicional para evitar la convergencia, en concreto, se trata de un mecanismo de reparación (ver sección 2.11.2).

Al igual que ϵ-MOGA, ϵ-GA utiliza las tres poblaciones $P(t)$,$A(t)$ y $G(t)$ (ver figura 5.3) cumpliendo la misma funcionalidad ya descrita en el capítulo anterior.

El pseudocódigo del algoritmo ϵ-GA es el siguiente:

```
1. t:=0
2. A(t):=∅
3. P(t):=ini_random(D)
4. eval(P(t))
5. A(t):=guardar(P(t),A(t))
6. modo:=exploración
7. while t<t_max do
8.      G(t):=crear(P(t),A(t))
9.      eval(G(t))
10.     A(t+1):=guardar(G(t),A(t))
11.     P(t+1):=actualizar(G(t),P(t))
12.     modo:=determinarmodo(P(t))
13.     t:=t+1
14. end while
```

Algoritmo 5.1: Algoritmo ϵ-MOGA.

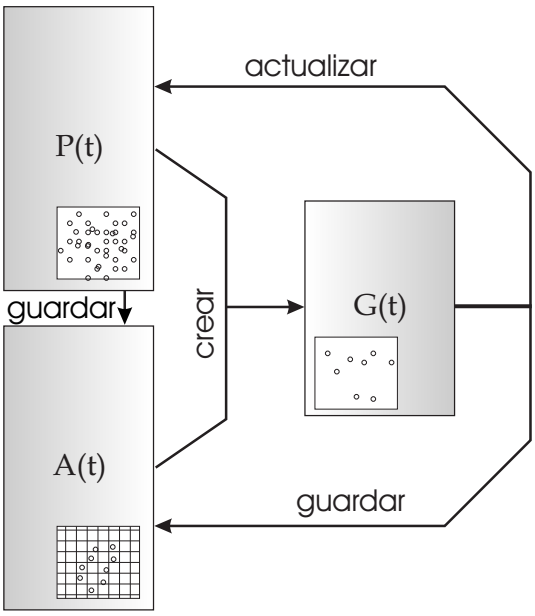

Figura 5.3: Estructura del algoritmo ϵ-GA.

A continuación se detallan los pasos más importantes del algoritmo:

- **Línea 3**. La población $P(0)$ es inicializada con $Nind_P$ individuos creados aleatoriamente dentro del espacio de búsqueda D.
- **Líneas 4 y 9**. La función *eval* calcula el valor de la función de *fitness* $J(\theta)$ para cada individuo θ de $P(t)$ (línea 4) o de $G(t)$ (línea 9) utilizando los 4 ordenadores de la plataforma multiordenador.
- **Línea 12**. La función *determinarmodo* establece el modo de funcionamiento del algoritmo de entre los modos *exploración* y *explotación*. Estos modos afectan a la manera en como se crean los nuevos individuos (función crear del algoritmo). El modo *explotación* se debe activar cuando se detecte que la población $P(t)$ ha convergido. Para ello, se utilizará una medida de la diversidad en el valor de la función de *fitness*. En concreto se tomará la diferencia entre el mejor valor

$$J_{P(t)}^{min} = \min_{\theta \in P(t)} J(\theta)$$

y el peor valor

$$J_{P(t)}^{max} = \max_{\theta \in P(t)} J(\theta)$$

en la iteración t. Si $J_{P(t)}^{max} - J_{P(t)}^{min} < \delta$ se activará el modo *explotación*[7], en caso contrario el modo será el de *exploración*. Inicialmente la **línea 6** se encarga de activar el modo *exploración*.

- **Líneas 5 y 10**. La función *guardar* analiza uno a uno los individuos de $P(t)$ (línea 5) o de $G(t)$ (línea 10) para determinar si serán incluidos en $A(t)$, para ello, el individuo tendrá que cumplir la condición de inclusión de la definición 5.8 eliminando los individuos que la misma definición especifica. Si al intentar introducir un individuo θ^1 en el archivo, su *box* ($\mathbf{box}(\theta^1)$) está ocupado por otro individuo θ^2 del archivo, es decir $\mathbf{box}(\theta^1) = \mathbf{box}(\theta^2)$, y $J(\theta^1) = J(\theta^2)$, se dejará en el archivo aquel individuo que más cerca esté del centro del *box* que ocupan[8]. De esta manera se cumple la condición de que cada *box* sólo puede ser ocupado por un individuo al mismo tiempo, y por otra parte, favorece una mejor distribución de las soluciones en el archivo.
- **Línea 8**. La función *crear* es la encargada de generar nuevos individuos y almacenarlos en la población $G(t)$ en cada iteración usando el siguiente procedimiento:
  1. Se seleccionan dos individuos aleatoriamente, θ^{p1} de $P(t)$, y θ^{p2} de $A(t)$.
  2. Si el algoritmo se encuentra en modo *exploración* θ^{p2} no se altera, mientras que si el modo es *explotación* el individuo es mutado, de la siguiente manera[9]:

$$\theta_i^{p2} = \theta_i^{p2} + N(0, \beta_{ini}).$$

[7] Si $J_{P(t)}^{min} = J^*$ todos los individuos en $P(t)$ serán soluciones globales quasi mínimas.

[8] Si esta situación es tratada estrictamente mediante la definición 5.8 conllevaría a la no inclusión del individuo θ^2.

[9] De esta manera se implementa el mecanismo de reparación.

3. Se lanza un número aleatorio $u \in [0 \dots 1]$. Si $u > P_{c/m}$ (probabilidad de cruce mutación) se realiza el paso 4 (cruce) sino el paso 5 (mutación).
4. θ^{p1} y θ^{p2} son cruzados mediante la técnica de recombinación lineal extendida generando dos individuos nuevos θ^{h1} y θ^{h2} de la siguiente manera ($\alpha_i(t)$ se determina igual que en el algoritmo ϵ-MOGA):

$$\theta_i^{h1} = \alpha_i(t) \cdot \theta_i^{p1} + (1 - \alpha_i(t)) \cdot \theta_i^{p2},$$

$$\theta_i^{h2} = (1 - \alpha_i(t)) \cdot \theta_i^{p1} + \alpha_i(t) \cdot \theta_i^{p2}.$$

5. θ^{p1} y θ^{p2} son mutados utilizando mutación aleatoria con distribución gaussiana ($\beta 1_i(t)$ y $\beta 2_i(t)$ se determinan igual que en el algoritmo ϵ-MOGA).

$$\theta_i^{h1} = \theta_i^{p1} + N(0, \beta 1_i(t)),$$
$$\theta_i^{h2} = \theta_i^{p2} + N(0, \beta 2_i(t)).$$

Este procedimiento se repite $Nind_G/2$ veces hasta que $G(t)$ esté llena.

- **Línea 11**. La función *actualizar* determina qué individuos de $G(t)$ serán incluidos en $P(t)$. Un individuo θ^G de $G(t)$ será insertado en $P(t)$ sustituyendo a θ^p, si $J(\theta^G) < J(\theta^p)$ siendo

$$\theta^p = \arg \operatorname*{máx}_{\theta \in P(t)} J(\theta)$$

de esta manera se asegura que el contenido en $P(t)$ va convergiendo.

Finalmente, cuando $t = t_{max}$, los individuos contenidos en el archivo $A(t)$ constituirán la solución al problema de optimización multimodal Θ_ϵ^*, considerándose Θ como el conjunto de individuos generados por las líneas de código 3 y 8, es decir,

$$\Theta = P(0) \bigcup \left(\bigcup_{0 \leq \tau < t_{max}-1} G(t) \right)$$

y considerando que $\Theta \cap \Theta^* \neq \emptyset$.

5.4. Evaluación del algoritmo

En esta sección, se pretende evaluar las prestaciones del algoritmo ϵ-GA utilizando una serie de problemas de optimización multimodales [20] modificados expresamente para tener infinitos óptimos globales. En el anexo A.2, se encuentran las funciones correspondientes a los cinco problemas que se utilizarán así como el conjunto de óptimos globales de cada uno de ellos.

Para cada uno de los problemas se comparan los resultados obtenidos por el algoritmo ϵ-GA y dos estrategias que ejecutan el algoritmo SQP desde puntos iniciales diferentes[10]. No se han utilizado otros algoritmos de optimización multimodales como los presentados en la sección 2.11 porque éstos no están preparados para optimizar funciones con infinitos óptimos globales. Tampoco se ha utilizado búsqueda exhaustiva o aleatoria porque sólo generarían resultados aceptables en espacios de búsqueda reducidos.

El primero de ellos, el $SQP_{aleatorio}$ ejecuta el algoritmo SQP partiendo desde un punto diferente cada vez, determinado aleatoriamente, dentro del espacio de búsqueda. En cada ejecución de dicho algoritmo se obtiene una solución que puede ser un mínimo local o global, descartándose los primeros. El algoritmo se detiene cuando se ha evaluado n_eval_max la función a optimizar[11].

El segundo, $SQP_{exhaustivo}$ es un algoritmo enumerativo que divide el espacio de búsqueda mediante un *grid* (con $n\_div - 1$ divisiones en cada dimensión) que abarca $(L-1)$ dimensiones de las L dimensiones del problema de optimización y lanza el algoritmo SQP sobre la dimensión que queda libre (reduciendo así el problema de optimización a $n\_div * (L-1)$ problemas unidimensionales). En cada uno de estos n_div problemas de optimización se lanzan SQPs desde puntos determinados aleatoriamente hasta que se evalúe n_div veces la función a optimizar. El parámetro n_div se determina de manera que $n\_eval\_max \cong n\_div^L$.

Los indicadores que se utilizan para analizar y comparar los resultados son: el número de soluciones quasi óptimas determinadas y el valor medio de la función a optimizar producido por éstas, distancia generacional[12], el espaciado y ratio *box* del conjunto de óptimos globales[13].

Dado que el proceso de optimización incorpora variables aleatorias, la optimización de cada problema, se repite 10 veces utilizando semillas diferentes, como se hizo en la evaluación del algoritmo ϵ-MOGA. Por lo tanto, se mostrarán los valores máximos, mínimos, la media y la varianza de cada indicador para las 10 optimizaciones diferentes que se lleven a cabo.

Para realizar la evaluación se han ajustado los siguientes parámetros de configuración del algoritmo ϵ-GA, que se mantienen constantes para cada uno de los problemas[14]:

- $Nind_G = 4$.
- $Nind_P = 100$.

[10] Es habitual en problemas de optimización global la utilización de algoritmos tipo *multi-start SQP* [66].

[11] n_eval_max es el número veces que se evalúa la función a optimizar tanto para ϵ-GA, $SQP_{exhaustivo}$ como $SQP_{aleatorio}$.

[12] Aplicada sobre el espacio de soluciones.

[13] Es el volumen del *box* formado por los extremos del conjunto de óptimos globales determinado por el algoritmo en cuestión, dividido por el volumen del *box* formado por los extremos del conjunto de óptimos globales real.

[14] Son los mismos que se utilizaron con el algoritmo ϵ-MOGA.

- t_{max} se elegirá en cada problema en concreto dependiendo de la dimensión y tamaño del espacio de soluciones.
- $P_{c/m} = 0.1$.
- $d_{ini} = 0.25$, $d_{fin} = 0.1$, $\beta_{ini} = 10.0$ y $\beta_{fin} = 0.1$.
- El parámetro $\delta(t)$ se va a ajustar de la siguiente manera para que sirva para todos los problemas de optimización

$$\delta(t) = \delta'(t) \cdot \bar{J},$$

donde:

- $\bar{J}$ representa la media de J para todos los individuos que han sido insertados en la población $P(t)$ durante el proceso de optimización. Con esto, lo que se está haciendo es una estimación de la media de la función de *fitness* J y así vincular el valor de δ al problema en cuestión[15].
- $\delta'(t)$ se determina mediante la siguiente expresión:

$$\delta'(t) = \frac{\delta_{ini}}{\sqrt{1 + \left(\left(\frac{\delta_{ini}}{\delta_{fin}}\right)^2 - 1\right) \frac{t}{(t_{max}-1)}}},$$

teniendo la precaución de que $\delta_{ini} \geq \delta_{fin}$ para que $\delta(t)$ vaya decreciendo a medida que el algoritmo evolucione.

Se utilizarán $\delta_{ini} = 0.1$ y $\delta_{fin} = 0.01$.

A continuación, se muestran los resultados obtenidos para cada problema con el algoritmo ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

- **Problema OP1**. Para este problema en concreto se utilizan los siguientes parámetros para el algoritmo ϵ-GA:
  - $\epsilon = 0.2$. Este valor ha sido escogido pequeño para poder caracterizar con precisión Θ^*.
  - $t_{max} = 100$ de manera que se evalúen 500 veces la función de test[16].

  La tabla 5.1 muestra los resultados de la optimización del problema OP1 con los algoritmos ϵ-GA y $SQP_{aleatorio}$, respectivamente. Sobre la tabla aparecen en negrita los mejores resultados obtenidos, en media, para cada indicador. Del análisis de los resultados se pueden obtener las siguientes conclusiones:

[15] Para asegurar que $\delta(t)$ nunca crezca sólo se tendrán en cuenta aquellos valores que se inserten en $P(t)$ con valor inferior a $\bar{J}$.

[16] El número de evaluaciones se determina como $n\_eval\_max = Nind_P + Nind_G * t_{max}$.

- El algoritmo ϵ-GA es claramente superior a $SQP_{aleatorio}$ en todos los indicadores menos en los relacionados con la convergencia, es decir, DG y $\bar{J}$ donde ambos algoritmos son iguales[17]. Además en 2 de las optimizaciones $SQP_{aleatorio}$ no ha alcanzado ninguna solución global mínima[18].
- El algoritmo ϵ-GA ha conseguido, en media, 7.6 soluciones globales mínimas ($\bar{J} = 0$ y $DG = 0$ así lo corroboran). Hay que tener en cuenta que con el ϵ escogido el máximo número de soluciones del archivo sólo podría ser de 8 (ver figura 5.4). Además con una caracterización de Θ^* muy buena ($RBOX = 0.93$) y una distribución buena ($SP \simeq 0$).

	$\lvert\Theta^*_\epsilon\rvert$	DG	$\bar{J}$	SP	RBOX
ϵ-MOGA					
mínimo	6	0	0	0.00077346	0.68918
máximo	8	0	0	0.0042677	0.99437
media	**7.6**	**0**	**0**	**0.0018064**	**0.93839**
std	0.69921	0	0	0.0012614	0.091623
$SQP_{aleatorio}$					
mínimo	1	0	0	0	0
máximo	3	0	0	0.14899	0.68017
media	1.875	**0**	**0**	0.019053	0.30732
std	0.83452	0	0	0.052514	0.30056

Tabla 5.1: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta^*_\epsilon\rvert$ (número de soluciones quasi óptimas), DG (distancia generacional), $\bar{J}$ (el valor medio de la función a optimizar producido por Θ^*_ϵ), SP (espaciado) y RBOX (ratio *box*) resultado de la optimización del problema OP1 con ϵ-GA y $SQP_{aleatorio}$.

La figura 5.4 muestra las soluciones globales quasi óptimas obtenidas por ambos algoritmos[19]. Si se observa esta figura se puede ver claramente que ϵ-GA caracteriza muy bien Θ^*.

- **Problema OP2**. Para este problema en concreto se utilizan los siguientes parámetros para el algoritmo ϵ-GA:
  - $\epsilon_1 = 0.2$ y $\epsilon_2 = 0.2$.
  - $t_{max} = 600$ de manera que se evalúen 2500 veces la función de test.

  La tabla 5.2 muestra los resultados de la optimización del problema OP2 con los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustiva}$, respectivamente. Del análisis de los resultados se pueden obtener las siguientes conclusiones:

[17] $J^* = 0$ para este problema. Los resultados del algoritmo $SQP_{aleatorio}$ con valor de $J > 0.01$ se ha eliminado por no considerarse soluciones globales mínimas.

[18] Estas optimizaciones no han sido tenidas en cuenta en el cálculo de los indicadores para no perjudicar los resultados.

[19] De las diferentes optimizaciones llevadas a cabo para cada uno de los algoritmos se escoge, para ser representada gráficamente, aquella que más cerca está de los valores medios obtenidos.

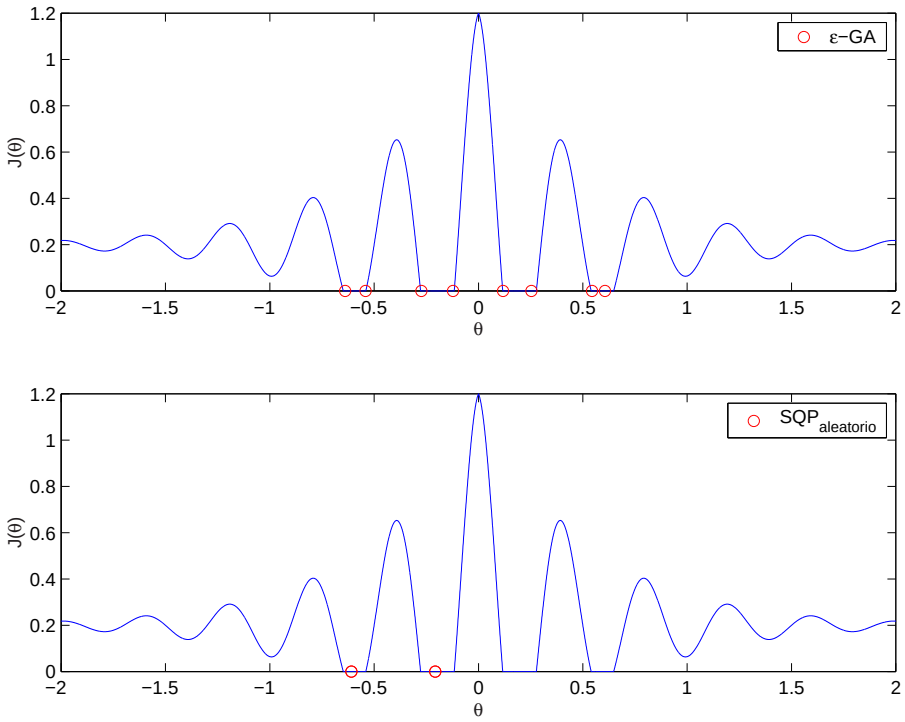

Figura 5.4: Conjunto de soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA y $SQP_{aleatorio}$ para OP1.

- El algoritmo ϵ-GA es bastante superior a $SQP_{aleatorio}$ y éste a $SQP_{exhaustivo}$ en número de soluciones quasi óptimas localizadas. Además ϵ-GA consigue caracterizar mejor Θ^* como indican los parámetros ($RBOX \approx 0.95$) y ($SP \simeq 0$).
- Los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ han conseguido mejor convergencia sobre Θ^* en las soluciones globales quasi óptimas que han alcanzado como indican $\bar{J} \simeq 0$[20] y $DG \simeq 0$. Evidentemente, estos algoritmos presentan mejores características, en cuanto a la precisión se refiere, a la hora de dar con soluciones óptimas. Para mejorar la precisión con ϵ-GA habría que ejecutar el algoritmo durante más tiempo.

La figura 5.5 muestra las soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$. Si se observa esta figura se puede ver claramente que de nuevo ϵ-GA ha caracterizado Θ^* mejor que los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

[20] $J^* = 0$ para este problema.

	$\lvert\Theta_\epsilon^*\rvert$	DG	$\bar{J}$	SP	RBOX
ϵ-MOGA					
mínimo	31	0.007065	0.15469	0.0034658	0.89772
máximo	39	0.01052	0.22258	0.014812	1.01
media	**36**	0.0088268	0.1803	**0.0060985**	**0.95873**
std	2.357	0.0013431	0.022672	0.0038822	0.036085
$SQP_{aleatorio}$					
mínimo	6	0.0002943	0.00097183	0.013591	0.39361
máximo	13	0.00042792	0.0033362	0.12565	0.98845
media	9.3	**0.00036243**	0.0017315	**0.059862**	0.71918
std	2.7909	4.5827e-5	0.00076741	0.036782	0.18277
$SQP_{exhaustivo}$					
mínimo	1	0.00016334	5.46e-005		0
máximo	2	0.00049319	0.0042821		0.037229
media	1.1429	0.00028694	**0.0012792**		0.0053185
std	0.37796	0.00014388	0.0016262		0.014071

Tabla 5.2: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_\epsilon^*\rvert$, DG, $\bar{J}$, SP y RBOX resultado de la optimización del problema OP2 con ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

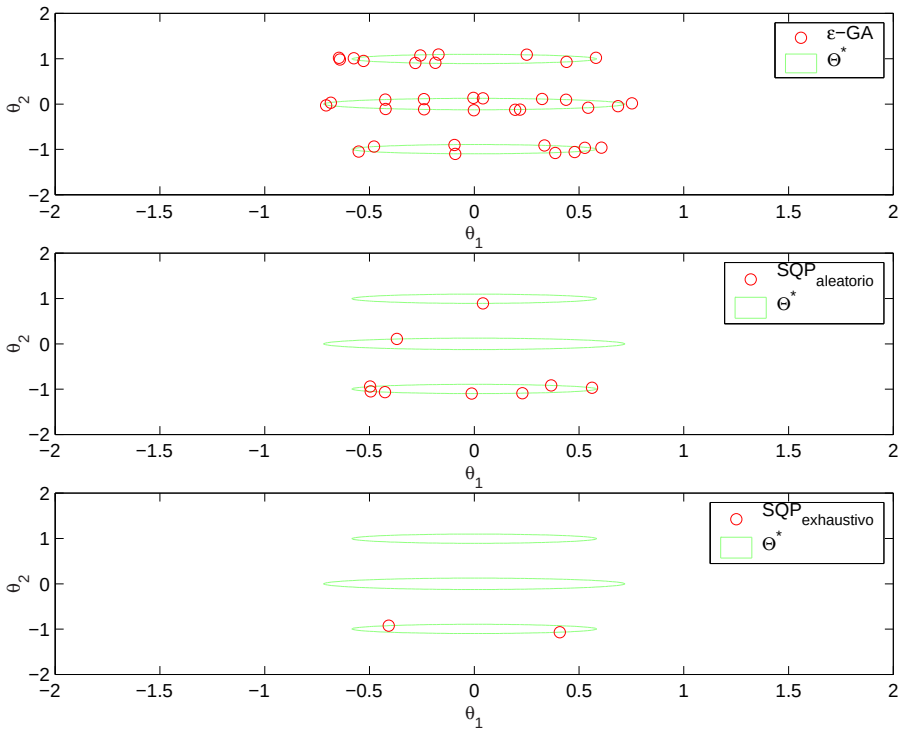

Figura 5.5: Conjunto de soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ para OP2.

- **Problema OP3**. Para este problema en concreto se utilizan los siguientes parámetros para el algoritmo ϵ-GA:

  - $\epsilon_1 = 0.65$ y $\epsilon_2 = 0.65$.
  - $t_{max} = 600$.

  La tabla 5.3 muestra los resultados de la optimización del problema OP3 con los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustiva}$, respectivamente. Del análisis de los resultados se pueden obtener las mismas conclusiones que para el problema OP2, es decir, ϵ-GA es bastante superior a $SQP_{aleatorio}$ y a $SQP_{exhaustivo}$ en número de soluciones quasi óptimas localizadas y consigue caracterizar mejor Θ^*. Los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$, en las soluciones globales quasi óptimas que han alcanzado, han conseguido mejor convergencia $\bar{J} \simeq 1$[21].

	$\lvert\Theta^*_\epsilon\rvert$	DG	$\bar{J}$	SP	RBOX
ϵ-MOGA					
mínimo	12	0.010269	1.0135	0.037929	0.98362
máximo	23	0.029081	1.0302	0.77488	1.0174
media	**16.3**	0.018971	1.0219	**0.2763**	**1.0022**
std	4.111	0.00683	0.0056346	0.26768	0.0091288
$SQP_{aleatorio}$					
mínimo	1	0.0014443	1	0	0
máximo	5	0.029138	1.0042	3.7542	0.94848
media	3	0.0056888	**1.002**	0.97247	0.288
std	1.4142	0.0084591	0.0012175	1.2515	0.37425
$SQP_{exhaustivo}$					
mínimo	1	0.0035499	1.0002		
máximo	1	0.003983	1.005		
media	1	**0.0036696**	**1.002**		
std	0	0.00013746	0.0015813		

Tabla 5.3: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta^*_\epsilon\rvert$, DG, $\bar{J}$, SP y RBOX resultado de la optimización del problema OP3 con ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

La figura 5.6 muestra las soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$. La figura muestra de nuevo como ϵ-GA ha conseguido caracterizar Θ^* de una manera más aceptable que los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

- **Problema OP4**. Para este problema en concreto se utilizan los siguientes parámetros para el algoritmo ϵ-GA:

  - $\epsilon_1 = 10$ y $\epsilon_2 = 10$.
  - $t_{max} = 600$.

[21] $J^* = 1$ para este problema.

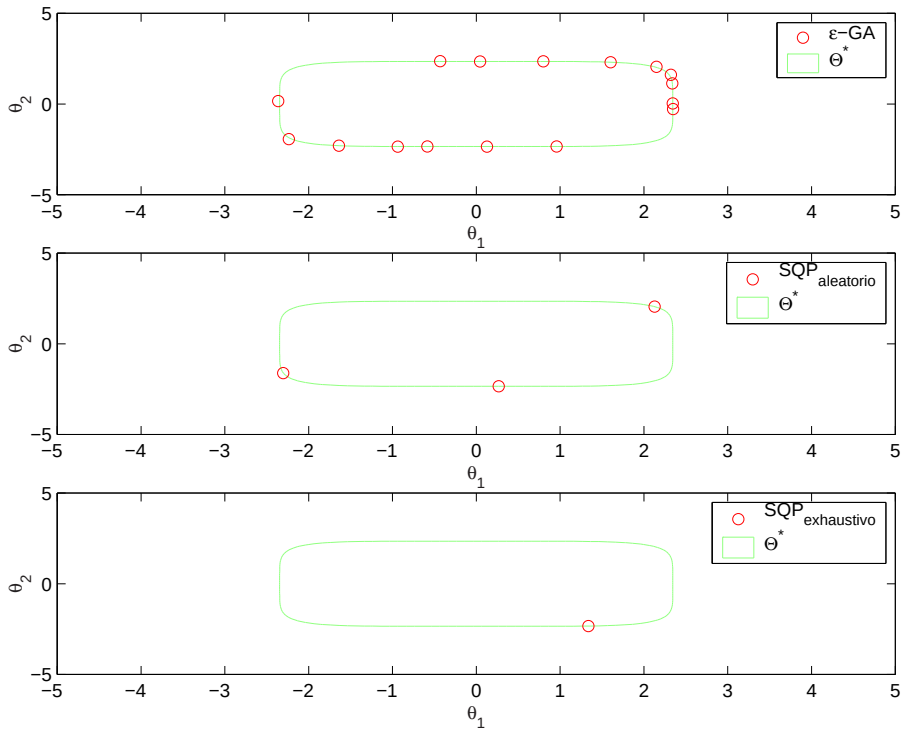

Figura 5.6: Conjunto de soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ para OP3.

La tabla 5.4 muestra los resultados de la optimización del problema OP4 con los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustiva}$, respectivamente. Del análisis de los resultados se pueden obtener las mismas conclusiones que para los problemas OP2 y OP3 pero más acentuadas, ya que, la dificultad de este problemas ha provocado:

- Que los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustiva}$ no consigan caracterizar Θ^*. De hecho en muchas de las optimizaciones no han conseguido localizar ninguna solución global mínima. ϵ-GA, aunque sí que ha podido caracterizar Θ^*, no le ha resultado fácil, de hecho, en una de las optimizaciones sólo ha alcanzado una solución global quasi mínima.
- Que la convergencia del algoritmo ϵ-GA no sea la más óptima como indican $\bar{J} \ncong 1$[22] y $DG \ncong 0$.

La figura 5.7 muestra las soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$. La figura muestra claramente como ϵ-GA ha conseguido caracterizar Θ^*, mientras que $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ no han podido.

[22] $J^* = 1$ para este problema.

	$\lvert\Theta_\epsilon^*\rvert$	DG	$\bar{J}$	SP	RBOX
ϵ-MOGA					
mínimo	1	0.43184	1.0024	0	0.015366
máximo	27	5.7393	1.028	27.644	0.98882
media	**21.889**	1.1225	1.0058	6.8674	**0.85217**
std	8.5212	1.7335	0.0083633	8.0181	0.3159
$SQP_{aleatorio}$					
mínimo	1	0.032309	1.0002	0	0
máximo	2	1.2967	1.0071	3.7542	0.00083188
media	1.2	**0.35126**	**1.0019**	**0.97247**	0.00016638
std	0.44721	0.54117	0.002973	1.2515	0.00037203
$SQP_{exhaustivo}$					
mínimo	1	0.45182	1.0026		
máximo	1	1.1113	1.0061		
media	1	0.88819	1.0048		
std	0	0.37794	0.0019861		

Tabla 5.4: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta_\epsilon^*\rvert$, DG, $\bar{J}$, SP y RBOX resultado de la optimización del problema OP4 con ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

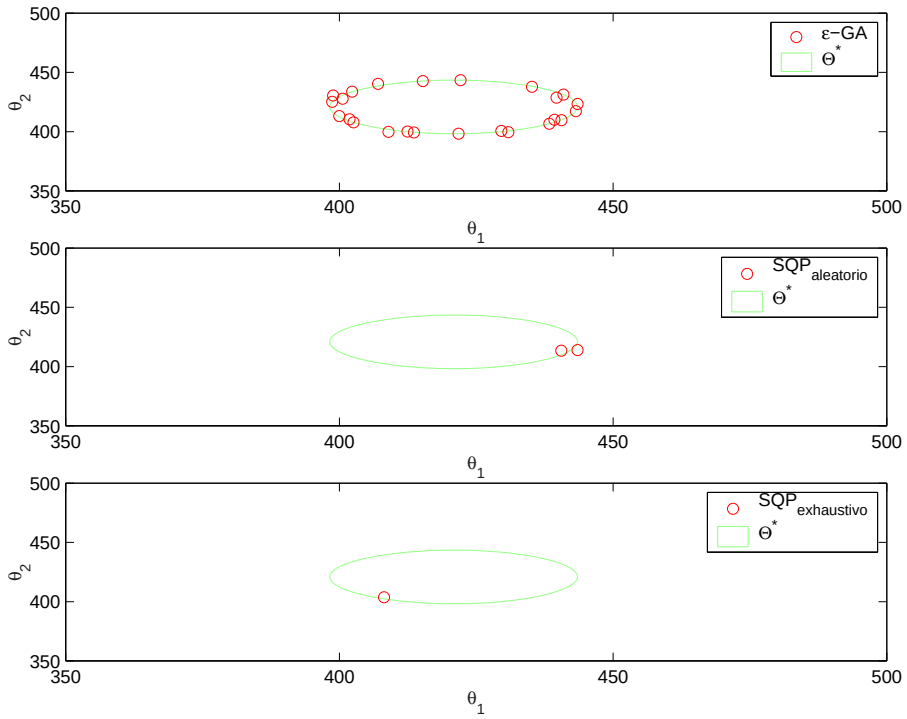

Figura 5.7: Conjunto de soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ para OP4.

- **Problema OP5**. Para este problema en concreto se utilizan los siguientes parámetros para el algoritmo ϵ-GA:

  - $\epsilon_1 = 10$, $\epsilon_2 = 10$ y $\epsilon_3 = 10$.
  - $t_{max} = 4975$ de manera que se evalúen 20000 veces la función de test.

  La tabla 5.5 muestra los resultados de la optimización del problema OP5 con los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustiva}$, respectivamente. Del análisis de los resultados se pueden obtener las mismas conclusiones que para el problema OP4, aún más acentuadas si cabe, ya que, ϵ-GA sí que ha conseguido caracterizar Θ^* pero $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ no lo han conseguido, de hecho, $SQP_{aleatorio}$ sólo ha conseguido detectar una solución global quasi mínima en una de las optimizaciones y $SQP_{aleatorio}$ en dos.

	$\lvert\Theta^*_\epsilon\rvert$	DG	$\bar{J}$	SP	RBOX
ϵ-MOGA					
mínimo	99	0.6738	1.0019	4.3571	0.92507
máximo	172	0.84915	1.0037	11.033	1.11
media	154.75	0.73087	1.0024	6.0202	1.0706
std	23.753	0.054206	0.00057957	2.1613	0.060324
$SQP_{aleatorio}$					
mínimo	2	0.50965	1.0032	0	0.033408
máximo	2	0.50965	1.0032	0	0.033408
media	2	0.50965	1.0032	0	0.033408
std	0	0	0	0	0
$SQP_{exhaustivo}$					
mínimo	1	0.78527	1.0047		0
máximo	2	1.4086	1.0097		0
media	1.5	1.0969	1.0072		0
std	0.70711	0.44076	0.0035139		0

Tabla 5.5: Valores mínimo, máximo, medio y desviación típica para los indicadores $\lvert\Theta^*_\epsilon\rvert$, DG, $\bar{J}$, SP y RBOX resultado de la optimización del problema OP5 con ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

La figura 5.8 muestra las soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$. Como ocurría con los problemas de optimización anteriores ϵ-GA ha conseguido caracterizar Θ^* de una manera más que aceptable, no siendo así para los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

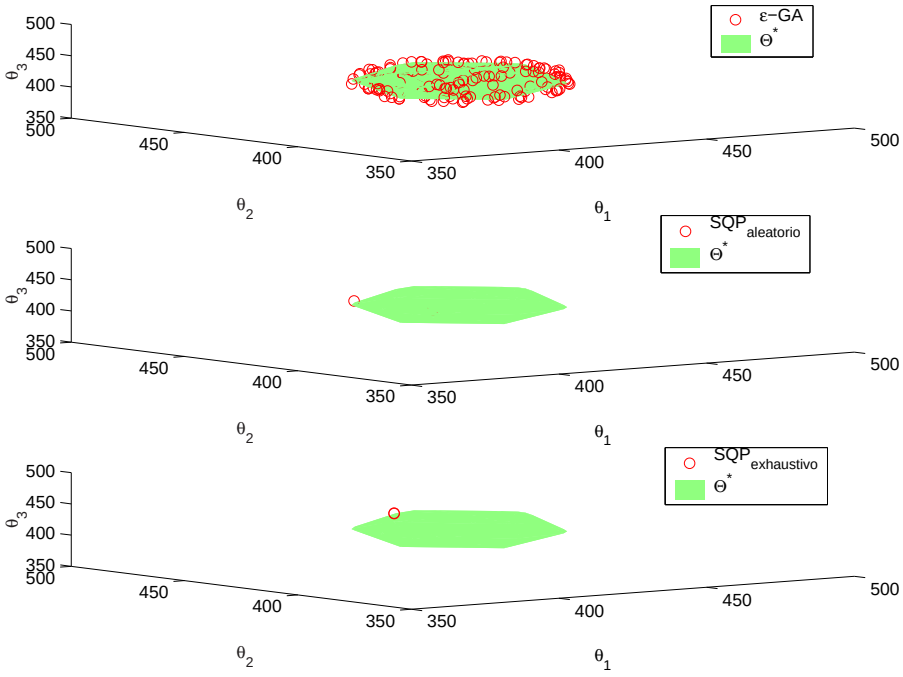

Figura 5.8: Conjunto de soluciones globales quasi óptimas obtenidas por los algoritmos ϵ-GA, $SQP_{aleatorio}$ y $SQP_{exhaustivo}$ para OP5.

5.5. Conclusiones

A lo largo de este capítulo se ha presentado un algoritmo genético llamado ϵ-GA, desarrollado especialmente para optimizar funciones multimodales con infinitos óptimos locales y globales. El algoritmo cumple los objetivos planteados al inicio del capítulo:

- Se trata de un algoritmo elitista, ya que no produce deterioro, como se ha demostrado y que, permite obtener como solución al problema de optimización, un conjunto Θ^*_ϵ que converge hacia el conjunto de óptimos globales Θ^*.
- Ofrece una buena distribución de las soluciones a lo largo del conjunto de soluciones globales mínimas que permite caracterizar Θ^* gracias al uso del *grid* y a la condición de que sólo una solución puede ser almacenada en un mismo *box* de dicho *grid*.
- Utiliza recursos de memoria limitados ya que, el número de individuos que puede contener el archivo está acotado.

Otras características de ϵ-GA son una arquitectura paralela maestro-esclavo, operadores de cruce y mutación con ajuste de parámetros basados en funciones y un mecanismo de reparación que permite evitar la convergencia prematura del algoritmo. El algoritmo ha demostrado, a través de la optimización de problemas especialmente complejos, como lo son los problemas OP1 a OP5, su buen funcionamiento comparado con los algoritmos $SQP_{aleatorio}$ y $SQP_{exhaustivo}$.

Los parámetros ϵ_i y δ juegan un papel importante en el funcionamiento del algoritmo y la elección de los mismos resulta crucial. El efecto de estos parámetros podría reducirse si se aplican algunas de las siguientes modificaciones:

- Adaptación de ϵ_i en función del contenido del archivo $A(t)$. De esta manera, se evitarían los problemas derivados de la elección de valores ϵ_i demasiado grandes (lo que podría acarrear una baja caracterización de Θ^*) o pequeños (más soluciones de las necesarias para caracterizar Θ^* con el sobrecoste computacional que esto supone). Por lo tanto, en vez de definir los parámetros ϵ_i, se definirían (como ya se hizo con $\not\epsilon$-MOGA) el número de divisiones del *grid* para cada dimensión.

  Con este algoritmo el inconveniente que se tendría es que no sería posible demostrar la convergencia de la solución de forma teórica.

- Otra posibilidad adicional, que permitiría mejorar la convergencia, es compaginar ϵ-GA con un algoritmo de optimización local (tipo SQP) que partiendo de la solución Θ^*_ϵ generada consiguiese mejorar la convergencia sobre Θ^*. Sería equivalente a hacer $\delta \simeq 0$.

Capítulo 6

Identificación Paramétrica de Sistemas

6.1. Introducción

Uno de los primeros pasos en muchas áreas tecnológicas es la obtención de un modelo matemático (modelado) que describa el comportamiento de un determinado sistema o proceso.

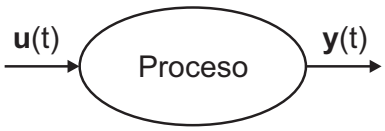

Figura 6.1: Representación de un proceso donde $\mathbf{y}(t)$ corresponde al vector de salidas y $\mathbf{u}(t)$ al vector de entradas.

En control de procesos, el modelado es un aspecto muy importante que, en parte, determinará la calidad del comportamiento final del bucle cerrado. Al proceso de modelado de la dinámica del sistema a partir de las observaciones de sus entradas y salidas, se le conoce con el nombre de identificación.

> *Identificación es la determinación, en base a las observaciones de las entradas y salidas del proceso, de un modelo dentro de una clase de modelos específicos, para el que el proceso en cuestión es equivalente.*
>
> L. Zadeh (1962)

La tarea de identificación de un proceso, por lo general, es una tarea costosa por el tiempo y recursos que emplea, de ahí que hayan sido muchos los esfuerzos llevados a cabo por los investigadores sobre las diferentes etapas y aspectos relacionados con dicho proceso. Una posible descripción del proceso de identificación podría ser la mostrada en la figura 6.2 donde, a partir del conocimiento previo que se tiene del proceso, se llevarían a cabo las siguientes etapas o fases [108]:

- **Diseño del experimento y preprocesado de los datos**. Esta etapa se centra en el diseño de las señales de entrada a aplicar sobre el proceso. El objetivo es excitar el sistema para poder observar, a través de sus salidas[1], su comportamiento dinámico. Si el comportamiento que se desea modelar no se observa en las salidas será imposible reflejarlo en el modelo. Tras realizar el experimento o experimentos sobre el proceso, por lo general, es necesario procesar los datos generados, por ejemplo mediante el empleo de filtros, para tratar de eliminar ciertos efectos indeseados de las señales como: ruido de medida, medidas erróneas (*outliers*), *offsets*, etc.
- **Seleccionar la estructura del modelo**. Se trata de definir el tipo de ecuaciones matemáticas que representarán el comportamiento del proceso. En este sentido es posible dividir los modelos en dos grupos [102]: los que contemplan los fenómenos

[1]Para ello se tomarán muestras de las salidas en cada intervalo de tiempo o periodo de muestreo.

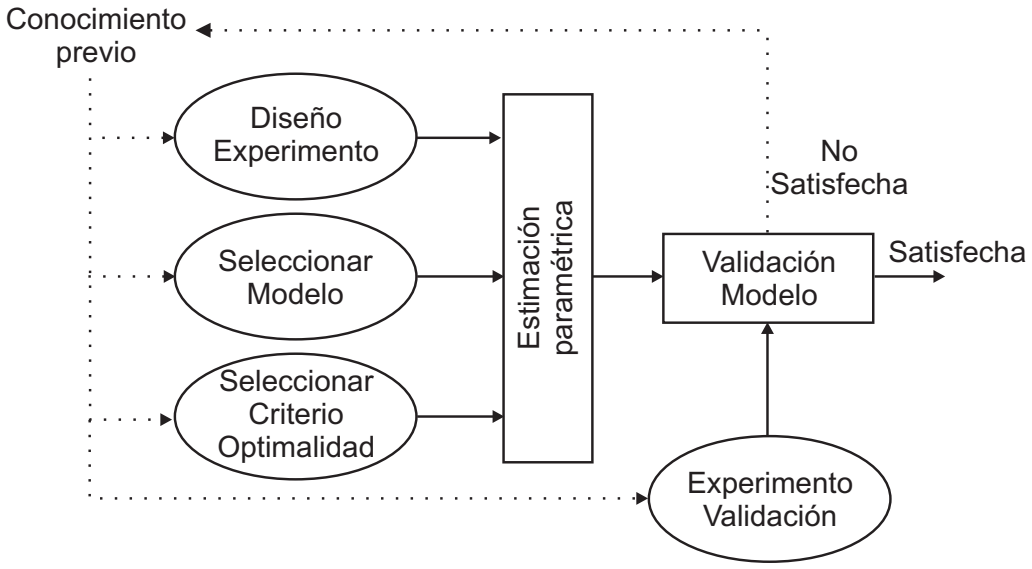

Figura 6.2: Etapas del proceso de identificación.

físicos y los que modelan el comportamiento del sistema sin atender a estos principios. El primer grupo suele utilizar ecuaciones en espacio de estados (conjunto de ecuaciones diferenciales o en diferencias). En estos modelos los parámetros tienen un significado físico y es posible utilizar información *a priori*, por ello, son conocidos como de primeros principios. El segundo grupo, por contra, intenta aproximar el comportamiento del proceso sin utilizar información *a priori*, por ejemplo, mediante polinomios, redes neuronales, conjuntos borrosos, etc., de ahí que sean conocidos como de caja negra.

La elección de una determinada estructura para el modelo establece las posibilidades de capturar cierta dinámica del proceso, por ejemplo, continuo o discreto, lineal o no, paramétrico o no, etc.

- **Determinar los parámetros del modelo a partir de los datos**. Se trata de obtener los parámetros desconocidos del modelo seleccionado utilizando para ello un algoritmo de estimación. En la mayoría de los casos estos algoritmos obtienen los valores de los parámetros a partir de la optimización de algún criterio. Cuando el modelo es lineal hay establecidas un conjunto de técnicas de identificación de éstos [90, 102], sin embargo, cuando los modelos son no lineales el procedimiento suele ser más complejo y particular.

- **Validar el modelo**. En esta última etapa se trata de confirmar la validez del modelo identificado. Para ello, se suele utilizar un conjunto de datos (entradas y salidas) distinto del utilizado en el proceso de identificación, para ver si el modelo reproduce el comportamiento descrito en dichos datos con una determinada precisión. Si no es así, el proceso de identificación debería ser repetido utilizando una estructura diferente para el modelo y/o datos diferentes hasta que se satisfaga el criterio de validación establecido.

En lo que a esta tesis concierne, los modelos a utilizar serán de primeros principios (lineales o no lineales sin ninguna limitación), continuos y se identificarán aquellos parámetros que sean desconocidos, por lo tanto, se tratará de identificación paramétrica. No se establecerá ninguna restricción en cuanto al criterio o criterios de optimalidad se refiere. Esto supone muchos grados de libertad para el diseñador, pero complica el proceso de optimización que, generalmente, será no convexo pudiendo presentar óptimos locales siendo necesario el uso de optimizadores globales. Es aquí donde los algoritmos evolutivos juegan un papel determinante como herramienta de optimización global, ya sea en su versión mono objetivo o multiobjetivo. El elevado coste computacional de los EAs, de momento, hace inviable la identificación en línea de la mayoría de procesos, sin embargo, fuera de línea resultarán muy adecuados como se comprobará en capítulos posteriores.

El proceso de identificación descrito determina lo que se conoce como el modelo nominal, sin embargo, un enfoque más general es aquel que persigue la obtención de un modelo nominal junto con su incertidumbre, provocada por la presencia de ruido de medida y/o dinámica no modelada. El modelo nominal y su incertidumbre constituirían el conjunto de modelos que serían considerados a la hora de realizar, por ejemplo, predicciones, diagnósticos, diseño de reguladores, etc.

El resto del capítulo incorpora los siguientes puntos: la sección 6.2 presenta el procedimiento a seguir en la identificación de los parámetros desconocidos del modelo no lineal a partir de los datos obtenidos a través de la experimentación sobre el proceso. Se describen también, diferentes criterios de optimalidad y sus características. Por último se citan trabajos relacionados con la identificación de procesos mediante algoritmos evolutivos. En la sección 6.3 se presentan los conceptos y formulación asociada a la identificación robusta (identificación de la incertidumbre). Se aborda, de forma especial, el caso de identificación robusta cuando el modelo del proceso es no lineal. Para terminar en la sección 6.4 se presentan las conclusiones más importantes.

6.2. Identificación paramétrica

La técnica está basada en la aceptación de una estructura inicial del modelo obtenida a partir del conocimiento *a priori* del mismo, del cual se desconocen los parámetros (o parte de ellos) pues el objetivo es determinarlos. Resulta habitual representar el modelo del proceso (lineal o no lineal) mediante un conjunto de ecuaciones diferenciales de primer orden que puede ser obtenido a partir de principios físicos.

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \mathbf{u}(t), \theta), \\ \hat{\mathbf{y}}(t, \theta) &= g(\mathbf{x}(t), \mathbf{u}(t), \theta). \end{aligned} \tag{6.1}$$

donde:

- $f(.), g(.)$ son las funciones del modelo.

- $\theta \in D \subset R^L$ es el vector[2] de parámetros desconocidos del modelo.
- $\mathbf{x}(t) \in R^n$ es el vector de estados del modelo.
- $\mathbf{u}(t) \in R^m$ es el vector de entradas del modelo.
- $\hat{\mathbf{y}}(t, \theta) \in R^l$ es el vector de salidas del modelo.

Se pretende conseguir que el comportamiento del modelo se ajuste lo mejor posible al del proceso real.

El comportamiento del proceso puede ser caracterizado mediante experimentos, en el caso de tratarse de procesos no lineales es aún más importante que los experimentos cubran todo el conjunto de comportamientos a modelar. Las señales de entrada podrían ser un conjunto de escalones a lo largo de todo el espacio hábil de las entradas sobre el cual se desea trabajar.

El comportamiento del modelo puede ser obtenido a partir de su simulación, aplicándole las mismas señales de entrada que se utilizaron en el experimento sobre el proceso. Los modelos a utilizar serán modelos continuos, de ahí que sea necesario utilizar métodos de integración numérica para su simulación.

Típicamente este objetivo se consigue mediante la minimización de una función donde se penalizan las diferencias, a lo largo del experimento, entre las salidas del proceso y el modelo (ver figura 6.3).

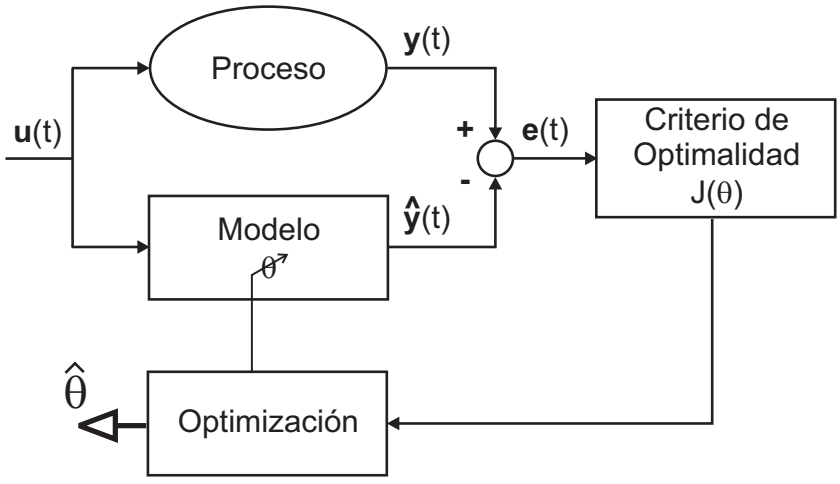

Figura 6.3: Problema de identificación paramétrica mediante optimización del criterio de optimalidad $J(\theta)$. $\hat{\theta}$ son los parámetros óptimos.

Definición 6.1 (Matriz Error de Identificación): *Se define la matriz del error de identificación* $\mathbf{E}(\theta)$ *de dimensiones* $(l \times N)$ *como*

[2] θ, $\mathbf{x}(t)$, $\mathbf{u}(t)$ e $\hat{\mathbf{y}}(t, \theta)$ son todos vectores columna.

$$\mathbf{E}(\theta) = \mathbf{Y} - \hat{\mathbf{Y}}(\theta),$$

donde:

- $\mathbf{Y} = [\mathbf{y}(t_1), \mathbf{y}(t_2) \dots \mathbf{y}(t_N)]$ *son las medidas de las salidas del proceso*[3] *consecuencia de la aplicación sobre el proceso de las entradas* $\mathbf{U} = [\mathbf{u}(t_1), \mathbf{u}(t_2) \dots \mathbf{u}(t_N)]$.
- $\hat{\mathbf{Y}}(\theta) = [\hat{\mathbf{y}}(t_1, \theta), \hat{\mathbf{y}}(t_2, \theta) \dots \hat{\mathbf{y}}(t_N, \theta)]$ *son las salidas simuladas del modelo*[4] *cuando se aplican las mismas señales* $\mathbf{U}$ *sobre sus entradas.*

△

La definición anterior arrastraría a su vez la definición de error de identificación asociado a una determinada salida.

Definición 6.2 (Vector Error de Identificación): *Se define el vector error de identificación* $\mathbf{e}_j(\theta)$ *para la salida* $j \in [1 \dots l]$ *como la fila* j *de la matriz error de identificación* $\mathbf{E}(\theta)$.

△

Y a la definición del error de identificación de una determinada salida y una determinada muestra.

Definición 6.3 (Error de Identificación): *Se define el error de identificación* $e_j(t_i, \theta)$ *para la salida* $j \in [1 \dots l]$ *y la muestra* $i \in [1 \dots N]$ *como el elemento* ji *la matriz error de identificación* $\mathbf{E}(\theta)$.

△

El error de identificación puede estar producido por errores sistemáticos (sesgo) y/o aleatorios (varianza). Los primeros son producidos, principalmente, por la dinámica no modelada, es decir, están relacionados con la estructura del modelo seleccionada, mientras que los segundos son producidos por la presencia de ruido de medida en los datos del experimento. En definitiva, sea cual sea la fuente o naturaleza del error de identificación su presencia evitará que las salidas del modelo reproduzcan exactamente a las del proceso.

Una vez se dispone de los datos y se ha seleccionado el modelo, tan sólo faltará por decidir la función a optimizar (criterio de optimalidad) que dará como solución los parámetros óptimos $\hat{\theta}$ del modelo para dicha función[5]. Una función comúnmente utilizada es la cuadrática que facilita la optimización en el caso de que los modelos sean lineales,

[3] $\mathbf{y}(t) \in \mathcal{R}^l$ es el vector columna de las salidas del proceso.

[4] Las salidas del modelo son calculadas integrando las ecuaciones (6.1). $t_i, i \in [1 \dots N]$ son los diferentes instantes de tiempo en los que son muestreadas las salidas del proceso y calculadas las del modelo. N es el número de muestras, de cada salida y entrada, del experimento. Se asume que el intervalo entre muestras es constantes $t_i = i \cdot T_s$, siendo T_s el periodo de muestreo o intervalo entre muestras.

[5] Por supuesto $\hat{\theta}$ dependerá del criterio de optimalidad escogido, de ahí que su elección deba ser claramente especificada y justificada.

pues la obtención de los parámetros es directa y no requiere métodos de optimización numéricos [102].

$$J(\theta) = \sum_{i=1}^{N} \sum_{j=1}^{l} k_{ji} e_j(t_i, \theta)^2 = \sum_{i=1}^{N} \sum_{j=1}^{l} k_{ji} (y_j(t_i) - \hat{y}_j(t_i, \theta))^2,$$

donde k_{ji} representa el coeficiente de ponderación de la salida j, para la muestra i.

El utilizar valores diferentes de ponderación para los k_{ji} tiene como objetivo:

- Poder eliminar ciertos datos utilizando $k_{ji} = 0$.
- Escalar el rango de las diferentes salidas.
- Dar mayor importancia a ciertas muestras de las salidas en ciertos instantes de tiempo, por ser puntos estratégicos en la respuesta del proceso (por ejemplo zonas de sobreoscilación, etc.).

La principal ventaja de utilizar funciones del tipo cuadrático se puede perder cuando el modelo es no lineal, ya que no estaría garantizada la obtención de los parámetros óptimos de forma explícita. Además la estimación de los parámetros puede resultar inadecuada cuando los datos presentan valores atípicos *outliers* ya que, su estimación se podría ver gravemente distorsionada por el efecto de los mismos [167]. Otro problema no menos importante es que las funciones cuadráticas distorsionan la influencia de los errores, ya que, los errores por debajo de la unidad son infravalorados, mientras que los que están por encima de la unidad son sobrevalorados.

Éste y otros motivos relacionados con las prestaciones que se desean para el criterio de optimalidad han hecho que aparezcan en la literatura otros criterios de optimalidad [167]. En cualquier caso, bien porque el modelo no sea lineal o porque el tipo de criterio de optimalidad que se utilice no sea cuadrático, los optimizadores tradicionales (por ejemplo, tipo SQP) pueden resultar inadecuados, ya que, $J(\theta)$ podría ser no convexa y/o presentar mínimos locales.

Los Algoritmos Evolutivos como herramientas de optimización global son una alternativa cada vez más empleada. Una de las ventajas de esta herramienta es que posibilita la resolución del problema con independencia del tipo de modelo y/o criterio de optimalidad que se plantee, pudiéndose de esta manera, definir la función $J(\theta)$ para que cubra los requisitos deseados sin tener, *a priori*, que cumplir ninguna condición.

6.2.1. Criterios de optimalidad

A continuación, en esta sección, se plantean diferentes enfoques a la hora de escoger el criterio de optimalidad, que definirá la función a optimizar $J(\theta)$. Estos enfoques se fundamentan, en mayor o menor medida, en la aceptación de ciertas hipótesis y en que,

en un principio, ningún enfoque tiene por qué ser mejor que otro. Los enfoques que se comentarán son los siguientes [102, 167]:

- Máxima probabilidad (*maximum likelihood*).
- Bayesiano.
- Robusto.

6.2.1.1. Máxima probabilidad

En este enfoque, se maximiza la siguiente función:

$$J(\theta) = \pi_Y(\mathbf{Y}|\theta), \tag{6.2}$$

que representa la densidad de probabilidad de $\mathbf{Y}$ cuando ésta es generada por un modelo con parámetros θ. Por lo tanto, este enfoque persigue la determinación de los parámetros $\hat{\theta}$ que hacen máxima la probabilidad de las medidas $\mathbf{Y}$.

$$\hat{\theta} = \arg \max_{\theta} \pi_Y(\mathbf{Y}|\theta) = \arg \max_{\theta} \ln(\pi_Y(\mathbf{Y}|\theta)). \tag{6.3}$$

Si se asume que no existen errores de modelado[6]

$$y(t_i) = \hat{y}(t_i, \hat{\theta}) + \mu(t_i)$$

y la salida del proceso está perturbada por un ruido $\mu(t_i)$ independiente cuya densidad de probabilidad es $\pi_{\mu(t_i)}(\mu(t_i))$ entonces

$$\pi_y(\mathbf{y}|\theta) = \prod_{i=1}^{N} \pi_{\mu(t_i)}(y(t_i) - \hat{y}(t_i, \theta))$$

y al aplicar logaritmo se transforma el productorio en sumatorio

$$\ln \pi_y(\mathbf{y}|\theta) = \sum_{i=1}^{N} \ln \pi_{\mu(t_i)}(y(t_i) - \hat{y}(t_i, \theta)). \tag{6.4}$$

Por lo tanto, la densidad de probabilidad de $\mu(t_i)$ determinará la función a optimizar. Por ejemplo, si se asume una distribución *Gaussiana* $\mathcal{N}(0, \sigma_{t_i}^2)$

$$\pi_{\mu(t_i)}(\mu(t_i)) = \frac{1}{2\pi\sigma_{t_i}^2} e^{-0.5(\frac{\mu(t_i)}{\sigma_{t_i}})^2}$$

y se aplica sobre (6.4) se puede llegar a los parámetros óptimos

[6]Por simplicidad se asume que el proceso sólo tiene una salida y.

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{N} \frac{(y(t_i) - \hat{y}(t_i, \theta))^2}{\sigma_{t_i}^2}, \tag{6.5}$$

que como se puede observar se obtienen de la minimización de una función cuadrática donde se utilizan coeficientes de ponderación $k_{1i} = 1/\sigma_{t_i}^2$ que se encargan de ponderar, en menor medida, aquellas medidas que estén más corrompidas. En el caso que $\sigma_{t_i} = \sigma \ \forall i$ la estimación $\hat{\theta}$ (obtenida mediante (6.5)) coincidiría con la que se obtendría de la minimización de la función cuadrática sin ponderación $k_{1i} = 1 \ \forall i$.

Este enfoque asume las siguientes hipótesis[7]:

- No existen errores de modelado.
- La influencia de las perturbaciones sobre las salidas puede ser modelada mediante señales aleatorias con distribución independiente, filtradas o no, por parte o todo el modelo.
- El número de muestras N de la salida debe ser elevado.

Cuando N tiende a ∞ se pueden establecer las siguientes propiedades del estimador de máxima probabilidad:

- Es consistente. Si θ^* son los parámetros del proceso,

$$\forall \delta > 0, prob(||\theta^* - \hat{\theta}|| \geq \delta) \to 0 \ \ cuando \ \ N \to \infty.$$

- Es asintóticamente eficiente. No existe ningún estimador consistente con menor covarianza cuando $N \to \infty$.

6.2.1.2. Bayesiano

Este enfoque considera una distribución de los parámetros θ, es decir, asume conocida, *a priori*, la densidad de probabilidad $\pi_\theta(\theta)$. De esta manera, aplicando la regla de *Bayes*, la probabilidad de densidad posterior de θ teniendo en cuenta las medidas de la salida $\mathbf{Y}$ vendría dada por

$$\pi_\theta(\theta|\mathbf{Y}) = \frac{\pi_{\mathbf{Y}}(\mathbf{Y}|\theta)\pi_\theta(\theta)}{\pi_{\mathbf{Y}}(\mathbf{Y})}. \tag{6.6}$$

Si se maximiza (6.6) respecto de θ (enfoque *maximum a posteriori*) y dado que $\pi_Y(\mathbf{Y})$ no depende de θ se tendría que

$$J(\theta) = \ln \pi_{\mathbf{Y}}(\mathbf{Y}|\theta)\pi_\theta(\theta) = \ln \pi_{\mathbf{Y}}(\mathbf{Y}|\theta) + \ln \pi_\theta(\theta).$$

[7]Se presentan las más relevantes, para un mayor detalle ver [102].

El primer término corresponde exactamente al enfoque de *máxima probabilidad* de la sección anterior, por lo tanto, este término se concreta teniendo en cuenta la información relacionada con la distribución del ruido. El segundo término lo que representa, como ya se ha comentado, es la información *a priori* que se tiene de los parámetros.

Por lo tanto, este enfoque que tiene el inconveniente de necesitar mayor información ($\pi_\theta(\theta)$), permite, al mismo tiempo, su incorporación cuando ésta está disponible. Esto hace que no sea necesario que los datos excedan el número de parámetros para asegurar unicidad. Por otra parte, mantiene las mismas hipótesis y propiedades que el enfoque de máxima probabilidad.

6.2.1.3. Robusto

Los enfoques vistos hasta el momento, asumen una serie de hipótesis que se alejan de la realidad, si se tiene en cuenta que los datos proceden de un sistema físico, por ejemplo, que no existan errores de modelado, que no se contemple la posibilidad de que las medidas puedan ser incorrectas debido a fallos en los sensores o que el error de identificación siga una determinada distribución (por ejemplo, normal o exponencial). Cuando estas hipótesis dejan de cumplirse podrían provocar la pérdida de prestaciones del estimador en mayor o menor medida. Cuando dicha pérdida de prestaciones no es considerable se dirá que el estimador es robusto [102].

Bajo este enfoque de estimador robusto, a continuación, se presentan dos métodos:

- $\mathcal{M}$-estimadores.
- Punto de corte.

$\mathcal{M}$-estimadores

Dentro de esta categoría estarían los estimadores que minimizan el siguiente criterio de optimalidad[8]:

$$J(\theta) = \sum_{i=1}^{N} \rho(e(t_i, \theta)). \tag{6.7}$$

donde $\rho(\cdot)$ es una función simétrica definida positiva con un único mínimo en cero.

La función de influencia (sensibilidades derivativas)

$$\psi(e(t_i, \theta)) = \frac{\partial \rho(e(t_i, \theta))}{\partial e(t_i, \theta)}$$

se encarga de medir la influencia de los datos sobre la estimación de los parámetros. El objetivo es plantear $\rho(\cdot)$ de manera que el estimador sea más robusto a medidas incorrectas

[8]Por simplicidad se asume, a lo largo de esta sección, que el proceso sólo tiene un salida.

que el propio estimador L_2 (mínimos cuadrados) que también es un $\mathcal{M}$-estimador donde $\rho(e(t_i,\theta)) = e(t_i,\theta)^2/2$ y $\psi(e(t_i,\theta)) = e(t_i,\theta)$, es decir, la influencia de los datos se incrementa linealmente con el error[9] lo que confirmaría la no robustez ante datos incorrectos pues, para éstos, $e(t_i,\theta)$ suele ser elevado.

La tabla 6.1 muestra las funciones de algunos de los $\mathcal{M}$-estimadores más comunes con sus funciones de influencia. La figura 6.4 muestra su representación gráfica.

Estimador	$\rho(e)$	$\psi(e)$
L_1	$\lvert e\rvert$	$sgn(e)$
$L_1 - L_2$	$2(\sqrt{1+e^2/2}-1)$	$\frac{e}{\sqrt{1+e^2/2}}$
L_p	$\frac{\lvert e\rvert^k}{k}$	$sgn(e)\lvert e\rvert^{k-1}$
Fair	$k^2\left(\frac{\lvert e\rvert}{k} - \log(1+\frac{\lvert e\rvert}{k})\right)$	$\frac{e}{1+\frac{\lvert e\rvert}{k}}$
Huber	$\begin{cases} e^2/2, & \lvert e\rvert \le k \\ k(\lvert e\rvert - k/2), & \lvert e\rvert \ge k \end{cases}$	$\begin{cases} e, & \lvert e\rvert \le k \\ k\cdot sgn(e), & \lvert e\rvert \ge k \end{cases}$
Tukey	$\begin{cases} \frac{k^2}{6}\left(1-(1-(\frac{e}{k})^2)^3\right), & \lvert e\rvert \le k \\ k^2/6, & \lvert e\rvert > k \end{cases}$	$\begin{cases} e\left(1-(\frac{e}{k})^2\right)^2, & \lvert e\rvert \le k \\ 0, & \lvert e\rvert > k \end{cases}$

Tabla 6.1: $\mathcal{M}$-estimadores con sus funciones de influencia.

El estimador L_1 reduce la influencia de las medidas incorrectas (errores elevados) comparado con el estimador L_2 aunque su función deja de ser estrictamente convexa. Otra propiedad del estimador L_1 (que no será estudiada aquí pero puede consultarse en [102]) hace que éste sea robusto ante las consideraciones que se hagan sobre la distribución del ruido. El estimador $L_1 - L_2$ tiene las ventajas de los estimadores L_1 y L_2, es decir, su función es convexa y reduce la influencia de las medidas incorrectas. El estimador L_p representa, en sí mismo, una familia de estimadores. En [163] se aconseja la utilización de $k = 1.2$ para obtener un buen estimador. El estimador $Fair$ tiene las derivadas de hasta tercer orden continuas y una solución única. Con $k = 1.3998$ se consigue en un 95 % la eficiencia asintótica cuando la distribución del error es normal. El estimador $Huber$ [87] es una parábola que incrementa linealmente a partir $|e| > k$. Con $k = 1.345$ se consigue en un 95 % la eficiencia asintótica cuando la distribución del error es normal. El estimador $Tukey$ es capaz de eliminar el efecto de las medidas incorrectas por completo, consiguiendo un 95 % de eficiencia asintótica con $k = 4.6851$. El problema de $Tukey$ es que no garantiza unicidad.

[9] Interesa que $\psi(e(t_i,\theta))$ esté acotada.

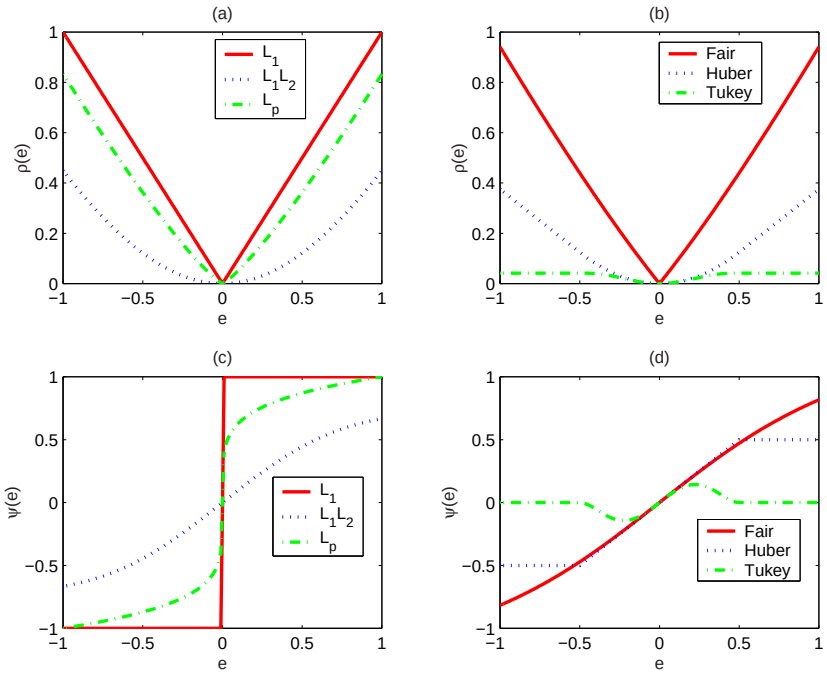

Figura 6.4: (a) y (b) representación de las funciones de los $\mathcal{M}$-estimadores de la tabla 6.1 $k = 1.2$ para L_p, $k = 1.3998$ para $Fair$ y $k = 0.5$ para $Huber$ y $Tukey$. (c) y (d) sus funciones de influencia.

Punto de corte

Estos estimadores son más robustos a las medidas incorrectas, ya que las eliminan, bien sea de forma implícita o explícita. Por ejemplo, el estimador de mínimos cuadrados de la mediana.

$$\hat{\theta} = \arg \min_{\theta} \operatorname*{med}_{i} e(t_i, \theta)^2. \tag{6.8}$$

Este estimador elimina de forma implícita el 50 % de los datos, lo que le hace muy robusto a las medidas incorrectas. Su inconveniente es que podría rechazar datos que abarcasen una parte de la respuesta con su consiguiente pérdida de identificabilidad. Por otra parte, la función mediana dificulta la optimización ya que no es posible encontrar una expresión directa de la misma.

Otra estrategia, conocida como diagnóstico de regresión, intenta iterativamente detectar los datos incorrectos para eliminarlos de la siguiente manera:

1. Determinar $\hat{\theta}$ utilizando todos los datos $\mathbf{y}$, mediante un estimador L_2 o L_1 por ejemplo.

2. Calcular el vector error de identificación para $\hat{\theta}$.

3. Eliminar aquellos datos, cuyo error de identificación, exceda un determinado umbral.

4. Si en el paso anterior no se ha eliminado ningún dato, el proceso se da por terminado. En caso contrario, se determina un nuevo $\hat{\theta}$ con los datos que quedan y se vuelve al punto 2.

Este procedimiento depende, en gran medida, de lo buena que sea la primera estimación que se realice de $\hat{\theta}$ y no garantiza una solución correcta aunque en la práctica funciona bien en identificaciones donde el número de datos incorrectos es moderado.

6.2.2. Identificación mediante EAs

Los algoritmos evolutivos han sido utilizados como herramientas de optimización en el proceso de identificación en multitud de trabajos y formas, especialmente cuando el modelo del proceso es no lineal y/o el criterio de optimalidad no es cuadrático. A continuación, se presenta una breve referencia bibliográfica como ejemplo.

En [137] se presenta la identificación de los parámetros de un propulsor de un vehículo aeroespacial mediante un GA y compara los resultados utilizando diferentes normas del error de identificación (norma-1, Fair, Huber y Turkey). El problema de optimización que se deriva de la identificación es multimodal y se muestra como el GA consigue obtener el mínimo global. Otro ejemplo similar se presenta en [155] donde se utilizan varios EAs, que utilizan mecanismos especiales para la optimización de funciones multimodales, en la identificación de 5 parámetros de un modelo no lineal de un motor de inducción.

Otros enfoques plantean la utilización de los EAs, no para la estimación de los parámetros del modelo sino de su estructura. Por ejemplo, en [57] se plantea la utilización de un GA en la determinación de los términos que debe incorporar la estructura de un modelo NARMAX (*Non-linear Auto-Regressive Moving Average with eXogenous inputs*). El objetivo que plantea es la minimización de la varianza del error de identificación y se aplica sobre una planta piloto donde la salida representa el nivel de un líquido. También se hace una extensión al enfoque multiobjetivo planteando dos criterios de optimalidad simultáneamente, la media y la varianza del error de identificación. Se utiliza el algoritmo MOGA como herramienta de optimización multiobjetivo.

Con este mismo planteamiento se han realizado muchos trabajos de identificación donde se utiliza una red neuronal y el EA se encarga de optimizar la estructura de dicha red, por ejemplo [65] y [52].

En la línea relacionada con la utilización de algoritmos evolutivos multiobjetivo, en [104] se presenta la identificación de 8 parámetros de un modelo de un vehículo utilizando SPEA2 para optimizar los 4 criterios de optimalidad que allí se proponen. Otro trabajo interesante es el que se muestra en [81] donde se utiliza un MOEA para la identificación de un sistema vibratorio que presenta ruido de alta frecuencia y retardo. El modelo tiene varias salidas y se aplica como criterio de optimalidad la norma-2 del espectro en frecuencia, para cada una de las salidas de forma independiente.

En los últimos años, desde la aparición de la programación genética (GP) aparece un enfoque nuevo más ambicioso para la identificación. El GP se utiliza simultáneamente para

determinar la expresión matemática del modelo (su estructura) y estimar sus parámetros [109]. Una posibilidad para componer la estructura del modelo, por ejemplo, se plantea interconectando operador matemáticos básicos formando una estructura en árbol, grafo, etc. Los elementos básicos podrían ser los operadores típicos (+,-,*,/) pero también debería contener no linealidades tipo zona muerta, saturaciones, retardos, etc.

En [156] se utiliza la programación genética en el proceso de identificación enfrentando criterios de optimalidad como prestaciones, complicación del modelo y criterio de validación. Para resolver este problema utiliza un MOGP. Evidentemente este enfoque tan ambicioso conlleva una carga computacional muy elevada. Para que sea abordable se suele limitar el número de operadores y el tamaño de la estructura del modelo.

Dentro de este contexto, identificación mediante EAs, el autor de la tesis ha realizado algunos trabajos encaminados al desarrollo de la misma. Por ejemplo, en [21] y [76] se muestra la identificación de los parámetros de un modelo no lineal (primeros principios) de un proceso térmico real utilizando un GA con codificación real como herramienta de optimización y utilizando la norma-1 del error de identificación como criterio de optimalidad. Con el mismo planteamiento en [79] se realiza la identificación de 15 parámetros del modelo no lineal de un invernadero (7 entradas y 2 salidas). Se utilizan coeficientes de ponderación para agrupar la norma-1 del error de identificación de cada una de las salidas en un mismo criterio de optimalidad. Este planteamiento tiene el inconveniente de tener que escoger los coeficientes de ponderación. En [113] se resuelve este problema planteando la identificación como un problema de optimización multiobjetivo que minimiza, de forma separada, el error de identificación de cada salida.

6.3. Identificación robusta

No siempre es suficiente con estimar los parámetros $\hat{\theta}$ que optimizan un determinado criterio de optimalidad (como los presentados en la sección 6.2.1, p.e.), en ocasiones, es necesario tener en cuenta la incertidumbre asociada a dichos parámetros. Uno de los casos más evidentes es la identificación para control robusto[10] (conocida también como identificación robusta) donde el control está basado en el modelo nominal del proceso y una adecuada incertidumbre asociada a dicho modelo nominal.

La incertidumbre en el modelo (en los parámetros del modelo en el caso de identificación paramétrica) puede ser causa, principalmente, del ruido asociado a las medidas tomadas sobre el proceso y a la dinámica no capturada por el modelo nominal (errores de modelado) debido a que el modelo siempre es una aproximación al proceso. Aunque la incertidumbre puede tener diferentes fuentes siempre se pondrá de manifiesto con la aparición del error de identificación.

A la hora de clasificar las diferentes técnicas de identificación robusta (IR) es posible establecer dos grandes categorías o enfoques [102, 139]:

[10] Ya sea como punto de partida para el diseño del controlador robusto o como mecanismo para establecer la predicciones que utilizará el controlador.

- **Estadístico**. Se asume que el error de identificación tiene ciertas propiedades estadísticas, por ejemplo, que éste puede ser modelado como una variable aleatoria con una determinada distribución, que en muchas ocasiones se supone también conocida. En este sentido, es posible utilizar la técnicas clásicas de identificación [108, 102] para determinar el modelo óptimo[11] $\hat{\theta}$ y su incertidumbre que vendrá determinada por las cotas de la matriz de covarianza de los parámetros estimados. Este enfoque podría ser cuestionable ya que, tratar los errores como variables aleatorias exclusivamente no considera la existencia de errores de modelado, lo cual no es posible cuando se identifica procesos reales, donde existen no linealidades, parámetros variables con el tiempo etc.

- **Determinístico**. Se asume que el error de identificación, aunque es desconocido, estará acotado. Este enfoque (como se plantea en [131, 159, 126]), a diferencia del anterior, puede resultar mucho más realista para la mayoría de casos prácticos, ya que, por ejemplo, los propios fabricantes suelen suministrar la precisión en la medidas de sus sensores mediante cotas máximas de error. Evidentemente, estas cotas deberían ser incrementadas en función de los errores estructurales que se asuman en el modelo (errores de modelado).

Este segundo enfoque, determinístico, resulta especialmente interesante cuando se trabaja con modelos no lineales de procesos de primeros principios, como en el caso de esta tesis, donde los parámetros del modelo tienen un significado físico y el modelo es una aproximación a la realidad. Evidentemente, este método tiene limitaciones como consecuencia de que la incertidumbre en los parámetros es sensible a las cotas que se determinen para el error de identificación, lo que puede llevar a la sobreestimación de las mismas y, por lo tanto, a la obtención de una incertidumbre excesivamente conservativa. Además, este enfoque presenta el inconveniente de que computacionalmente puede resultar bastante complejo y costoso.

Dado que este será el enfoque a utilizar en el resto de la tesis, todos estos problemas serán comentados en detalle en las secciones posteriores.

6.3.1. Conjunto de parámetros factibles (FPS)

Si se opta por un enfoque determinístico[12], como es el caso de esta tesis, no se asume, en un principio, ninguna distribución para el error de identificación, por contra, se asume que se conoce una cota sobre él, ya sea sobre la integral del vector del error de identificación (principalmente norma-1 o norma-2) o sobre el error de identificación en cada muestra (norma-∞ sobre el vector de identificación).

En cualquier caso, la solución al problema de identificación robusta será el conjunto de parámetros que mantienen el error de identificación acotado con respecto a la norma

[11] Utilizando, por ejemplo, un enfoque de máxima probabilidad o bayesiano.

[12] En la literatura se conoce también como *Bounded error data* o *Set Membership Estimation*.

que se establezca. Este conjunto de parámetros se conoce como conjunto de parámetros factibles (FPS)[13].

Definición 6.4 (Conjunto de Parámetros Factibles): *Dado un espacio $D \subseteq \mathcal{R}^L$, se define el conjunto de parámetros factibles (FPS) como:*

$$FPS := \{\theta \in D : \|\mathbf{E}(\theta)\|_p^w \leq \eta, \eta > 0\}, \tag{6.9}$$

donde $||\cdot||_p^w$ representa la norma p con pesos w y η la cota para dicha norma.

$\triangle$

Las normas más utilizadas, como ya se ha comentado, son la norma-1, la norma-2 y la norma-∞, aunque, en un caso más general, se podría utilizar cualquier otra norma o expresión relacionada con θ. Evidentemente, tanto la norma utilizada como su cota condicionará el FPS, pudiendo resultar en un $FPS = \emptyset$. En [9] se presenta una comparativa entre diferentes normas. La norma sobre el error de identificación que se utilice estará condicionando el conjunto de incertidumbre en las salidas.

Definición 6.5 (Conjunto de Incertidumbre en las Salidas): *Dado un espacio $\Lambda \subseteq \mathcal{R}^{l \times N}$, se define el conjunto de incertidumbre en las salidas (MUS) como:*

$$MUS := \{\mathbf{Y} \in \Lambda : \|\mathbf{E}(\theta)\|_p^w \leq \eta, \eta > 0\}, \tag{6.10}$$

donde $||\cdot||_p^w$ representa la norma p con pesos w y η la cota para dicha norma.

$\triangle$

De ambas definiciones se deduce que $\|\mathbf{E}(FPS)\|_p^w \subseteq MUS$. La figura 6.5 esboza la relación entre estos conjuntos y variables para el caso donde $L = 2$ (modelo con dos parámetros a identificar) $N = 3$ (se dispone de tres muestras de la salida del proceso $l = 1$) y se aplica la norma-∞.

La determinación exacta del FPS no resulta, en general, sencilla, ya que, el FPS podría ser un conjunto no convexo e incluso inconexo, de ahí que, ciertas técnicas de IR se hayan limitado, simplemente, a aproximar el FPS. El FPS está estrechamente relacionado con la norma del error de identificación que se utilice (norma-∞ si no se especifica alguna otra en particular) y el modelo del proceso. Cuando éste es lineal respecto de sus parámetros y se aplica la norma-2 sobre el error de identificación, el FPS es un elipsoide cuyo centro es la estimación de mínimos cuadrados [61], mientras que si se utiliza la norma-∞ el FPS es un politopo convexo en D más o menos complicado, no pudiéndose vaticinar su forma, en un principio, para sistemas no lineales y/o otras normas.

Por ejemplo, si se utiliza el siguiente modelo lineal

$$\hat{y}(t_i) = [u(t_i - 1)\ y(t_i - 1)] \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \Phi(t_i)\theta$$

[13] *Feasible Parameter Set*. Cuando la identificación no es paramétrica sería el conjunto de modelos factibles (*Feasible System Set (FSS)*).

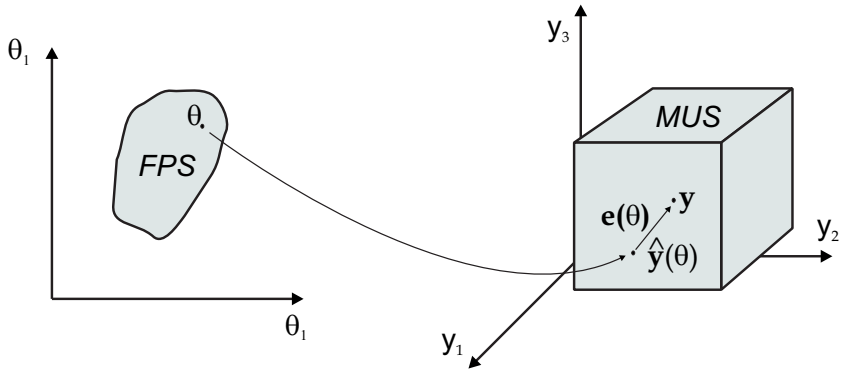

Figura 6.5: FPS y MUS (zonas sombreadas) para un caso particular donde $\mathbf{y} \in \mathcal{R}^3$ y $\theta \in \mathcal{R}^2$. Se ha utilizado una norma infinito, por eso, MUS es un cubo con centro en la medida $\mathbf{y} = [y(t_1), y(t_2), y(t_3)]^T$.

y se acota el error de identificación $|e(t_i, \theta)| \leq e_{max}$, cada medida que se tome de la salida limitará el FPS a estar contenido dentro de la franja delimitada por las siguientes inecuaciones

$$y(t_i) - e_{max} \leq \Phi(t_i)\theta,$$

$$y(t_i) + e_{max} \geq \Phi(t_i)\theta,$$

por lo tanto, a mayor número de medida, más complicada podrá ser su forma (ver figura 6.6).

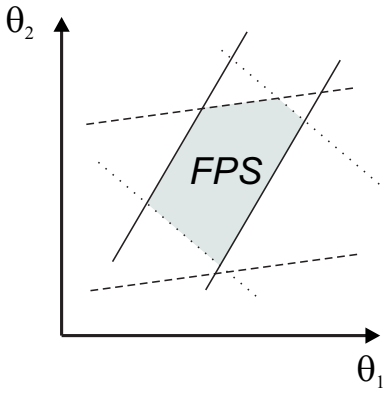

Figura 6.6: Ejemplo de un FPS (zona sombreada) para un modelo lineal de dos parámetros cuando se toman tres medidas de su salida.

Resulta interesante la determinación exacta del FPS, ya que como se apunta en [9]:

- Podría revelar en qué medida se podría reducir su tamaño mediante una mejor planificación del experimento.

- Se podría utilizar para determinar la distancia máxima (error de estimación de los parámetros) entre dos puntos pertenecientes al FPS.
- Podría ser utilizado para determinar el centro del FPS (modelo de peor caso).

Existen en la bibliografía diferentes formas de describir el FPS, por ejemplo:

- Mediante el conjunto de todos sus vertices adecuadamente enumerados e indicando, para cada uno de ellos, sus vertices adyacentes [127].
- Mediante el conjunto de todas sus caras N-dimensionales [25].
- Mediante un conjunto de aristas y los hiperplanos que las soportan [158].

Sin embargo, la dificultad que acarrea determinar el FPS exacto ha provocado que muchos algoritmos de IR se limiten única y exclusivamente a aproximar el FPS mediante formas más sencillas como elipsoides [55], ortotopos [17] o paralelotopos [29].

La figura 6.7 muestra como sería la aproximación del FPS de la figura 6.6 mediante una elipse y un rectángulo de volumen mínimo. Se puede observar como al aproximar el FPS (ya sea mediante elipses o rectángulos) el conjunto resultante FPS' contendrá al FPS. De ahí que considerar FPS' siempre supondrá un situación más conservativa.

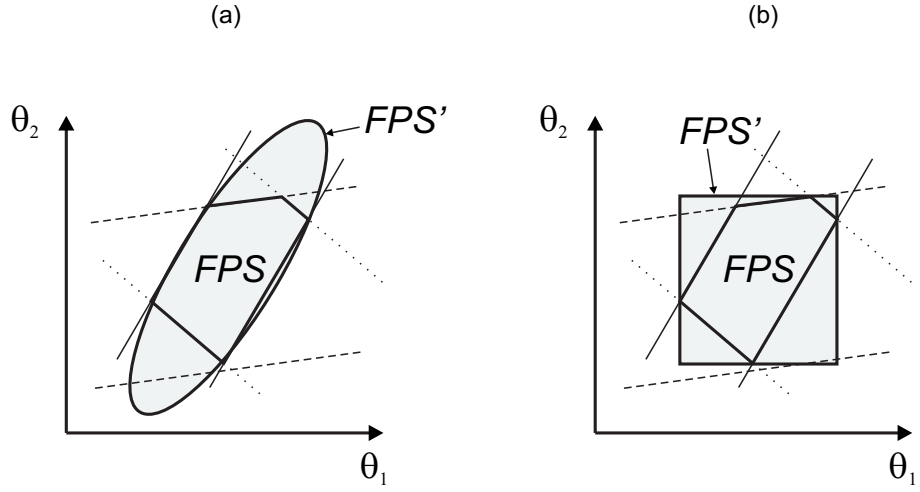

Figura 6.7: FPS' es la aproximación del FPS mediante (a) elipse de volumen mínimo y (b) rectángulo de volumen mínimo.

Cuando el modelo es no lineal respecto de sus parámetros el FPS podría ser no convexo e incluso inconexo debido a que el FPS no tiene por qué estar limitado por rectas paralelas (en el caso N dimensional hiperplanos) pudiendo ser curvas (hipersuperficies en el caso N dimensional). A continuación, se muestran dos ejemplos para ilustrar esta situación [96]:

1. Situación donde el FPS es no convexo utilizando un modelo exponencial con dos parámetros.

$$\hat{y}(t_i) = \theta_2 e^{-\theta_1 t_i}. \tag{6.11}$$

Cada medida que se toma de la salida genera dos inecuaciones del tipo

$$\theta_2 \geq (y(t_i) - e_{max})e^{\theta_1 t_i},$$
$$\theta_2 \leq (y(t_i) + e_{max})e^{\theta_1 t_i}.$$

2. Situación donde el FPS discontinuo y además inconexo. Se utiliza un modelo senoidal con dos parámetros también.

$$\hat{y}(t_i) = \sin(\theta_1 t_i) + \theta_2. \tag{6.12}$$

Cada medida que se toma de la salida genera dos inecuaciones del tipo

$$\theta_2 \geq y(t_i) - e_{max} + \sin(\theta_1 t_i),$$
$$\theta_2 \leq y(t_i) + e_{max} + \sin(\theta_1 t_i).$$

La figura 6.8 muestra el FPS para el modelo exponencial de la ecuación (6.11) y senoidal (ecuación (6.12)) cuando se toman dos medidas de la salida. En ambos casos el FPS es no convexo y para el modelo senoidal, además, es inconexo.

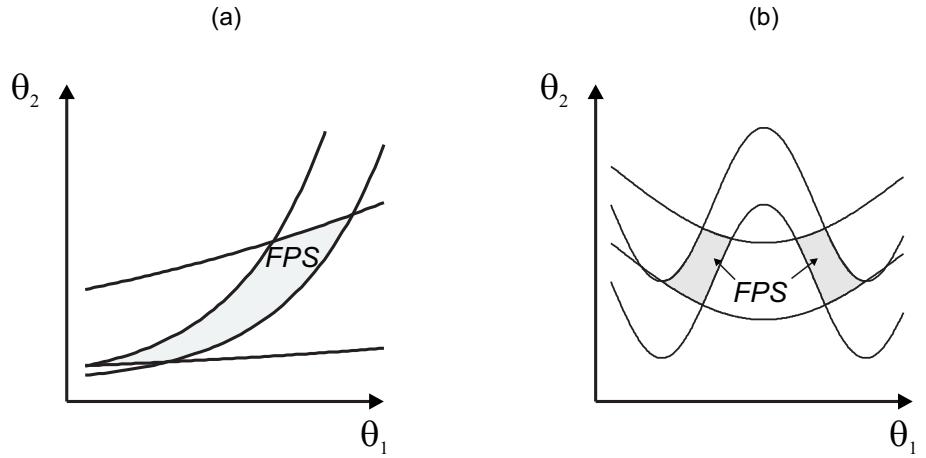

Figura 6.8: (a) FPS no convexo generado por el modelo de la ecuación (6.11) cuando se toman dos medidas de la salida. (b) FPS no convexo e inconexo generado por el modelo de la ecuación (6.12) cuando se toman dos medidas de la salida.

6.3.2. Identificación robusta en sistemas no lineales

Una vez planteada la problemática asociada a la IR cuando el modelo es no lineal (el FPS puede ser no convexo e incluso inconexo) se comentan en esta sección las técnicas de IR no lineal más comunes:

- Enfoque de IR lineal.
- Programación Sigmoidal.
- Nube de puntos.
- Contorno del FPS.
- Análisis intervalar.

6.3.2.1. Enfoque de IR lineal

Bajo este enfoque coexisten todas aquellas técnicas que son extensión de métodos utilizados en IR cuando el modelo es lineal.

El procedimiento más evidente pasa por aplicar las técnicas de IR lineal sobre el modelo linealizado alrededor de un determinado valor de los parámetros a estimar. En [18] se utiliza un proceso iterativo que determina, utilizando programación lineal y múltiples linealizaciones del modelo, los intervalos de incertidumbre para cada uno de los parámetros (ortotopo). Este procedimiento es costoso, desde un punto de vista computacional, ya que tiene que resolver múltiples problemas de programación lineal y depende de la linealización del modelo y entorno al que punto en que se realice.

Otra alternativa [132], válida para modelos con una determinada estructura, OE[14] por ejemplo, redefine el modelo del error para obtener un conjunto de inecuaciones lineales que han de ser satisfechas por todos los $\theta \in FPS$.

6.3.2.2. Programación sigmoidal

En este caso se pretende obtener un ortotopo que contenga al FPS. Para ello, se plantean $2L$ problemas de optimización para determinar las cotas inferiores (minimizando) y superiores (maximizando) de cada parámetro,

$$J(\theta) = \theta_i,\ i = 1, \ldots, L$$

sujeto a

$$g_j(\theta) \geq 0, i = 1, \ldots, 2N.$$

[14] *Output error model.* Se trata de un modelo lineal respecto de sus entradas y no lineal respecto de sus parámetros debido a la recursividad.

Si $g_j(\theta)$ se puede expresar de forma explícita mediante una función polinomial, el problema puede resolverse mediante programación sigmoidal [125]. Esta técnica se ha aplicado a modelos no lineales discretos en espacio de estados, ARMA y multi-exponenciales [159]. Como inconveniente hay que comentar que el problema sigmoidal es no convexo y puede generar soluciones locales. Además puede necesitar la reparametrización del modelo.

6.3.2.3. Nube de puntos

Bajo esta categoría se encuentran aquellas técnicas que generan una nube de puntos dentro del espacio de búsqueda que pertenecen al FPS o a su contorno.

En [51] se explora aleatoriamente el espacio de búsqueda con un algoritmo del tipo Monte Carlo y se van almacenando todos los puntos $\theta \in FPS$ que se encuentran. La técnica de búsqueda aleatoria puede ser combinada con el análisis de componentes principales a la hora de caracterizar la nube de puntos [95]. Estos métodos son bastante potentes, ya que no presentan limitaciones en cuanto al tipo de modelo, norma aplicada sobre el error de identificación y forma del FPS. Sin embargo, computacionalmente son bastante ineficientes, especialmente cuando el espacio de búsqueda es grande.

6.3.2.4. Contorno del FPS

Este enfoque es similar al planteado para la nube de puntos pero, en este caso, los puntos a localizar son aquellos que pertenecen al contorno del FPS. Si $\|\mathbf{E}(\theta)\|_p^w$ es continua para los puntos del contorno del FPS, el contorno en sí se puede definir de la siguiente manera:

Definición 6.6 (Contorno del FPS, ∂FPS): *Dado un espacio $D \subseteq \mathcal{R}^L$, se define el contorno del conjunto de parámetros factibles (∂FPS) como:*

$$\partial FPS := \{\theta \in D : \|\mathbf{E}(\theta)\|_p^w = \eta, \eta > 0\}, \tag{6.13}$$

donde $||\cdot||_p^w$ representa la norma p con pesos w y η la cota para dicha norma.

△

En [133] se plantea un procedimiento determinístico, cuando el modelo es diferenciable respecto de sus parámetros, para dar con un conjunto de puntos que pertenecen al ∂FPS. Para ello, primero se detecta un punto inicial $\theta_0 \in \partial FPS$ y a partir de éste se va recorriendo el contorno dando saltos de tamaño variable en función de su curvatura. El inconveniente de este método, aparte de que el modelo tiene que ser diferenciable respecto de sus parámetros, es que no resulta adecuado para espacios de búsqueda de dimensión mayor que tres y cuando el FPS es inconexo.

Otra alternativa, conocida como OMNE (*Outlier Minimal Number Estimator*) se planteada en [103] cuando $FPS \neq \emptyset$. Esta técnica tiene dos pasos:

1. Determinar un punto $\theta_0 \in FPS$. Para ello se utiliza un optimizador global que maximiza $J(\theta)$ igual al porcentaje del n^o de las inecuaciones (que describen el FPS) satisfechas

$$\theta_0 = \arg\max J(\theta).$$

2. A partir de θ_0 se traza una recta en una determinada dirección $\vec{d}$ y se busca el corte de dicha recta con el ∂FPS utilizando para ello, la optimización dicotómica. De esta manera, modificando la dirección $\vec{d}$ se van obteniendo diferentes puntos sobre el ∂FPS.

Este método es bastante eficiente, aunque presenta algunos inconvenientes. El coste computacional es elevado y el resultado puede depender, en gran medida, del punto inicial θ_0 que se utilice, ya que, si la recta corta más de una vez el contorno sólo se detectaría uno de los cortes, situación que puede fácilmente darse cuando el FPS es no convexo y que se da siempre que el FPS sea inconexo (ver figura 6.9).

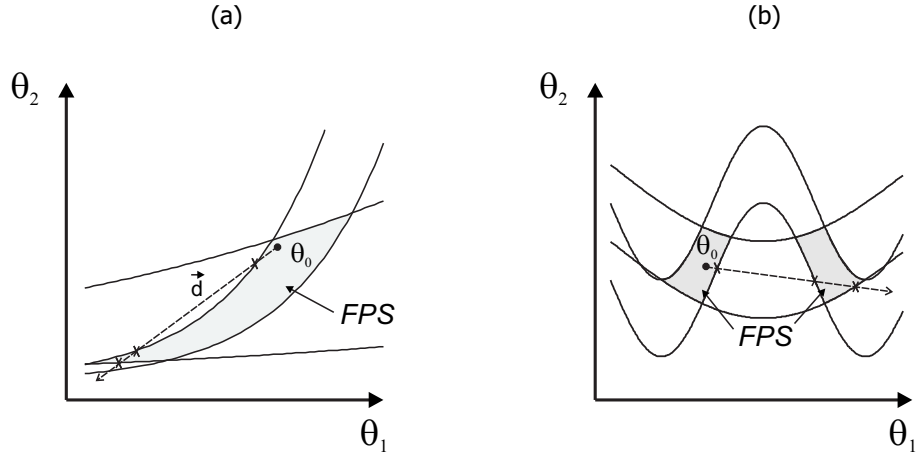

Figura 6.9: Situación donde la recta $\vec{d}$ corta con el ∂FPS más de una vez. (a) FPS no convexo, se podrían detectar hasta tres cortes de $\vec{d}$ con el ∂FPS en función de θ_0. (b) FPS inconexo, independientemente de θ_0 en muchas direcciones se producirían varios cortes con el ∂FPS.

Otra alternativa, que aproxima el ∂FPS mediante una resolución iterativa de un problema de regresión de vectores soporte se presenta en [96]. Para conseguir aproximar el ∂FPS con cierta precisión divide el espacio de búsqueda mediante un conjunto de puntos distribuidos uniformemente, sobre los que se aplica dicho procedimiento. Dado que este enfoque no genera una buena aproximación en espacios de búsqueda grandes y/o ∂FPS pequeños, se plantea ir dividiendo secuencialmente el espacio de búsqueda (o aquellas zonas donde se desea una mayor precisión). La determinación del ∂FPS se realiza de forma local aproximando los vectores (o superficies) de regresión, mediante expansiones de Taylor de primer orden. La técnica se restringe a la aplicación de la norma infinito y no resulta muy adecuada cuando el ∂FPS no es alisado.

6.3.2.5. Análisis intervalar

Este método caracteriza el FPS, cuando el vector de error de identificación $|e(\theta)| \leq e_{max}$, mediante la unión de un conjunto de ortotopos (*boxes*)

$$FPS' = \bigcup_i [\theta]_i.$$

Para ello, utiliza el algoritmo recursivo SIVIA (*Set Inverter Via Interval Analysis*) [102, 157] que analiza, por medio de una función de inclusión $[f]([\theta])$, si un determinado (*box*) $[\theta]$ pertenece o no al FPS. Para ello, $[f]([\theta])$ tiene que pertenecer al MUS que será un *box* en el espacio de las salidas.

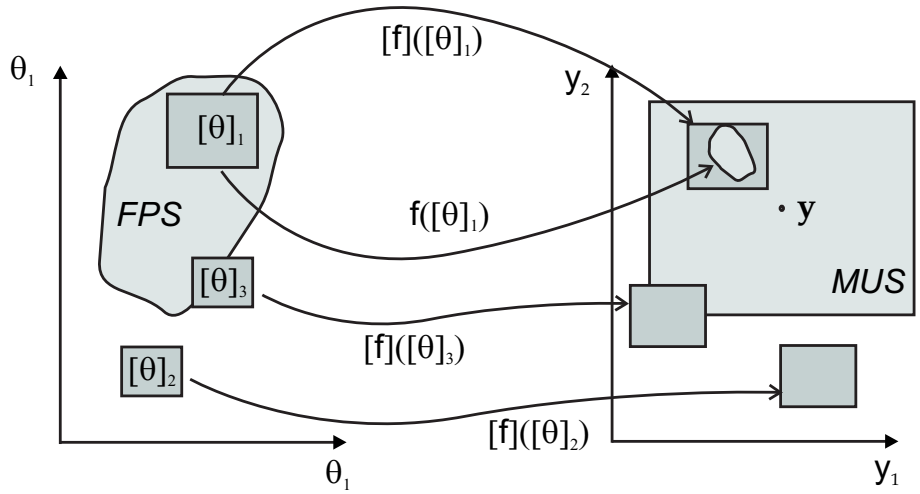

Figura 6.10: *Boxes* y sus imágenes obtenidas mediante la función de inclusión f. Dado que $[f]([\theta]_1) \subset MUS \Rightarrow [\theta]_1 \subset FPS$, mientras que no ocurre lo mismo con $[\theta]_2, [\theta]_3 \nsubseteq FPS$, ya que, $[f]([\theta]_2), [f]([\theta]_3) \nsubseteq MUS$.

Interesa que el *box* producido por $[f]([\theta])$, además de asegurar que $[f]([\theta]) \supseteq f([\theta])$, tenga el menor volumen posible, problema que aún no está resuelto cuando la función es compleja. Esto provoca que este método sea conservativo debido a la propagación del error que se produce cuando $[f]([\theta]) \supset f([\theta])$. Además la forma en la que se calcule también puede afectar. Sirva el siguiente ejemplo para ponerlo de manifiesto. Si se analizan estas tres funciones:

$$f_1(\theta) = \theta(\theta + 2),$$
$$f_2(\theta) = \theta^2 + 2\theta,$$
$$f_3(\theta) = (\theta + 1)^2 - 1,$$

cuando $[\theta] = [-1, 1]$, todas ellas darían $f([\theta]) = [-1, 3]$, sin embargo, si se aplican las operaciones básicas de intervalos sobre ellas el resultado no es siempre el mismo:

$$[f_1]([\theta]) = [\theta]([\theta] + 2) = [-3, 3],$$

$$[f_2]([\theta]) = [\theta]^2 + 2[\theta] = [-2, 3],$$

$$[f_3]([\theta]) = ([\theta] + 1)^2 - 1 = [-1, 3],$$

es decir, sólo en el tercer caso se consigue $[f_3]([\theta]) = f([\theta])$. En los otros dos casos los intervalos obtenidos son más conservativos, fenómeno que se ve aumentado cuando las funciones son complejas. Esto, junto con la complicación de obtener las funciones de inclusión, hace que esta técnica, aunque elegante, sea muchas veces inviable.

6.3.3. Modelo nominal

La dificultad que puede entrañar la determinación del FPS, hace que, en ocasiones, el problema de IR suela limitarse a la determinación de un modelo nominal $\hat{\theta}$ que:

- Pertenezca al FPS, $\hat{\theta} \in FPS$.
- Sea óptimo bajo algún criterio de optimalidad.

Si bien ambos objetivos, obtención del FPS y del $\hat{\theta}$, no son excluyentes, sino todo lo contrario. El modelo nominal por excelencia es el centro de Chebyshev del FPS [60]

$$\hat{\theta}_c = \arg \min_{\theta \in D} \max_{\overline{\theta} \in FPS} \|\theta - \overline{\theta}\|. \tag{6.14}$$

Este modelo, minimiza la máxima distancia entre θ_c y el vector de parámetros desconocidos que generaría los datos reales bajo la hipótesis que éste pertenece al FPS. θ_c presenta una serie de características:

- Es un modelo óptimo en términos del error del estimación en los parámetros, pero no en términos del error de identificación.
- Resulta sensible, al igual que la determinación del FPS, a la cota η que se establezca para el error de identificación, como es evidente.
- Si el FPS es inconexo, podría ocurrir que $\theta_c \notin FPS$. De ahí que se prefiera determinar el centro de Chebyshev condicionado, θ_{cc} como:

$$\hat{\theta}_{cc} = \arg \min_{\theta \in FPS} \max_{\overline{\theta} \in FPS} \|\theta - \overline{\theta}\|. \tag{6.15}$$

Otra posibilidad, a la hora de determinar el modelo nominal que sí es óptima en términos del error de identificación y menos sensible a η, sería utilizar un estimador de proyección [60] para obtener:

$$\hat{\theta}_p = \arg \min_{\theta \in FPS} ||\mathbf{e}(\theta)||. \tag{6.16}$$

El más estudiado y utilizado, sin duda, sería el estimador de mínimos cuadrados L_2 cuando se minimiza la norma-2 o los estimadores de proyección correspondientes L_1 o L_∞ cuando se utiliza la norma-1 o la norma-∞ respectivamente. Estas estimaciones serían de máxima probabilidad, por ejemplo, $\hat{\theta}_{\ell_2}$ lo sería cuando el error de identificación presenta una distribución gaussiana, $\hat{\theta}_{\ell_1}$ cuando la distribución es Laplaciana y $\hat{\theta}_{\ell_\infty}$ cuando la distribución es uniforme. Además, $\hat{\theta}_{\ell_1}$ y $\hat{\theta}_{\ell_\infty}$ son más robustos respecto a la incertidumbre en el conocimiento del tipo de distribución [126].

En [10] se presenta otra posibilidad de obtención del modelo nominal utilizando un estimador de proyección interpolada, que se conoce como centro analítico:

$$\hat{\theta}_a = \arg \max_{\theta \in FPS} \sum_{i=1}^{N} \ln(\eta^2 - e(t_i, \theta)^2), \tag{6.17}$$

que también produce una estimación de máxima probabilidad cuando la distribución del error de identificación es parabólica o una aproximación truncada de la distribución gaussiana.

6.4. Conclusiones

En este capítulo se han descrito, en líneas generales, las etapas asociadas a la identificación de procesos, haciendo hincapié, en la etapa de la identificación de los parámetros (identificación paramétrica) de modelos no lineales de primeros principios fuera de línea.

En esta etapa de identificación paramétrica, los parámetros desconocidos del modelo se obtienen a través de la optimización de un determinado criterio de optimalidad, con la intención de que el modelo se ajuste al comportamiento del proceso, reflejado mediante los datos de entrada/salida obtenidos a través de la experimentación sobre este último.

En el proceso de identificación se pueden dar una serie de errores (diferencia entre el comportamiento del modelo y del proceso) producidos por la existencia de ruido de medida (con la posibilidad de que hayan fallos en las medidas) y/o errores de modelado, que condicionarán el resultado. Estos errores, así como las propiedades de los parámetros identificados se ven influenciados, en gran medida, por el criterio de optimalidad que se utilice. Por ello, se ha dedicado una sección exclusivamente a analizar diferentes enfoques a la hora de escoger el criterio de optimalidad, entre ellos, el de máxima probabilidad y el bayesiano (con propiedades de consistencia y eficiencia bajo el cumplimiento de ciertas hipótesis) y el robusto (a posibles fallos en las medidas).

El uso de modelos no lineales y la flexibilidad de poder escoger un determinado criterio de optimalidad conlleva un aumento en la dificultad asociada a la optimización de éste, ya que se pueden plantear problemas de optimización multimodales, lo que requeriría la utilización de optimizadores globales como por ejemplo los Algoritmos Evolutivos. La potencia y facilidad de uso de los EAs permite que se puedan aunar todos los esfuerzos en definir el criterio de optimalidad, para que cubra los requisitos deseados, y no en su optimización.

Un enfoque de identificación robusta, persigue la determinación, no sólo de los parámetros que hacen óptimo un determinado criterio, sino también la incertidumbre asociado a los mismos. En este sentido, dos enfoques pueden ser utilizados: el estadístico (cuando se asumen ciertas propiedades estadísticas del error de identificación) o el determinístico (cuando se asume que el error de identificación, aunque desconocido, está acotado).

Este segundo enfoque resulta más realista, especialmente cuando se trabaja con modelos no lineales de procesos de primeros principios, como es el caso de esta tesis. El objetivo es determinar el FPS (conjunto de parámetros que mantiene el error de identificación acotado) que para sistemas lineales resulta ser un politopo convexo que suele ser aproximado por elipsoides o ortotopos, etc. Cuando el modelo es no lineal el FPS puede ser no convexo y/o inconexo dificultando su obtención. En este caso existen diferentes procedimientos para obtener el FPS cada uno con sus correspondientes ventajas e inconvenientes. Algunos de ellos:

- Son conservativos porque aproximan el FPS por un ortotopo, por ejemplo.
- Son aplicables sólo cuando el modelo presenta una estructura determinada o es sencillo.
- Son muy flexibles pero computacionalmente ineficientes y aplicables sólo a $FPS \in \mathcal{R}^2$ o $FPS \in \mathcal{R}^3$.
- Presentan limitaciones relacionadas con la forma del FPS, cuando son inconexos o su contorno no es alisado.

De ahí que no exista una procedimiento o técnica única, siempre pujando éstas entre flexibilidad y coste computacional. En el siguiente capítulo se presentará una metodología novedosa, en cuanto a la técnica y computación se refiere, basada en los algoritmos evolutivos $\not\epsilon^2$MOGA y ϵ-GA que permitirá determinar el FPS con la máxima flexibilidad posible y mostrando eficiencia desde el punto de vista computacional.

Capítulo 7

Identificación Robusta de Sistemas no Lineales mediante ϵ-GA y $\not\epsilon^{\nearrow}$-MOGA

7.1. Introducción

El objetivo de este capítulo es mostrar una metodología flexible y computacionalmente eficiente, basada en los algoritmos $\not\epsilon^2$MOGA y ϵ-GA presentados en los capítulos anteriores, que permita caracterizar el conjunto de parámetros factibles FPS cuando se plantea un problema de identificación robusta (IR) paramétrica. Con esta metodología se puede:

1. Utilizar cualquier tipo de modelo paramétrico para el proceso, siempre y cuando sus salidas puedan calcularse mediante simulación.
2. Acotar el error de identificación mediante diferentes normas simultáneamente sin restricciones sobre el tipo de norma utilizada.

Para ello, se plantea un problema de optimización multimodal donde los mínimos globales de la función a optimizar representan el ∂FPS[1] y los modelos del FPS constituyen soluciones globales quasi mínimas. Dado que podrán existir infinitas soluciones mínimas y quasi mínimas globales, se procederá a la obtención de un conjunto finito de éstas FPS^* que sirva para caracterizar el FPS y en especial el ∂FPS. Para ello se utilizará el algoritmo de optimización ϵ-GA y la metodología tendrá las siguientes características:

- Se podrán caracterizar FPS de cualquier tipo: convexos, no convexos e inconexos.
- Será un enfoque no conservativo ya que el FPS no se aproximará mediante ortotopos, elipsoides, etc.
- Permitirá calcular, fácilmente, una buena aproximación al modelo de peor caso $\hat{\theta}_c$, centro de Chebyshev del FPS.

Adicionalmente, con el objetivo de determinar otros modelos nominales, aparte del centro de Chebyshev, se plantea la obtención de los modelos óptimos con respecto al error de identificación utilizando estimadores de proyección. Al utilizar varias normas del error de identificación simultáneamente se plantea un problema de optimización multiobjetivo donde las soluciones óptimas de Pareto (respecto a las diferentes normas) constituyen los modelos nominales óptimos de proyección. Para obtener dichos modelos óptimos se utiliza el algoritmo $\not\epsilon^2$MOGA. Los modelos nominales óptimos servirán, como se mostrará a lo largo del capítulo, como referencia para determinar las cotas mínimas a aplicar sobre las diferentes normas del error de identificación y el espacio de búsqueda del FPS.

El resto del capítulo se desarrolla de la siguiente manera: la sección 7.2 se encarga de describir matemáticamente el escenario de IR planteado a lo largo de esta introducción, en la sección 7.3 se presenta un ejemplo de IR de un sistema térmico que servirá para ilustrar la técnica, a continuación en la sección 7.4 se aborda la IR de un proceso biomédico y para terminar en la sección 7.5 se detallan las conclusiones del método de IR presentado y del capítulo.

[1] Contorno del conjunto de parámetros factibles.

7.2. Formulación matemática

El escenario a tener en cuenta contempla que el error de identificación pueda estar acotado por varias normas[2] simultáneamente.

Por ello, se define la siguiente norma del error de identificación

Definición 7.1 (Norma N_i del error de identificación): *Se define N_i como la norma p, ponderada por los coeficientes* $\mathbf{w}$*, sobre el vector del error de identificación para la salida j de la siguiente manera:*

$$N_i(\theta) = \|\mathbf{e}_j(\theta)\|_p^{\mathbf{w}} \ , \quad i \in A := [1, 2, \ldots, s], \tag{7.1}$$

donde s es el número de normas que se utilizaran en la IR.

△

Por lo tanto, existirá un FPS_i consistente con cada norma N_i y cota η_i y su contorno

Definición 7.2 (FPS_i): *Dado un espacio $D \subset \mathcal{R}^L$, se define el conjunto de parámetros factibles (FPS_i) para la norma N_i y la cota η_i como:*

$$FPS_i := \{\theta \in D : N_i(\theta) \leq \eta_i, \eta_i > 0\}. \tag{7.2}$$

△

Definición 7.3 (Contorno del FPS_i, ∂FPS_i): *Dado un espacio $D \subseteq \mathcal{R}^L$, se define el contorno del conjunto de parámetros factibles (∂FPS_i) como:*

$$\partial FPS_i := \{\theta \in D : N_i(\theta) = \eta_i, \eta_i > 0\}. \tag{7.3}$$

△

Y por lo tanto el conjunto de parámetros factibles para todas las normas simultáneamente se obtendrá como:

$$FPS := \{\bigcap_{i \in A} FPS_i\} = \{\theta \in D : \forall i \in A, N_i(\theta) \leq \eta_i, \eta_i > 0\}. \tag{7.4}$$

Si $N_i(\theta)$ $\forall i$ es continua en los puntos del contorno del conjunto de parámetros factibles, dicho contorno se puede definir de la siguiente manera:

Definición 7.4 (∂FPS): *Dado un espacio $D \subset \mathcal{R}^L$, se define el contorno del conjunto de parámetros factibles (∂FPS) consistente con las normas N_i y cotas η_i como:*

$$\partial FPS := \{\theta \in D : \exists i | N_i(\theta) = \eta_i \wedge N_j(\theta) \leq \eta_j, \ \eta_i, \eta_j > 0, \ i, j \in A\}. \tag{7.5}$$

△

[2]En un caso más general podrían establecerse cotas sobre funciones cualesquiera y no tener porque utilizar exclusivamente normas.

Para caracterizar el FPS se plantean, a continuación, dos alternativas:

A) A través de la caracterización del ∂FPS. Para ello se crea una función $J(\theta)$ de manera que sus mínimos globales constituyan el ∂FPS. *A posteriori* se puede caracterizar el FPS tomando un conjunto finito de puntos dentro del ∂FPS. Esta solución será válida para el caso en el que $N_i(\theta)$ sea continua en el contorno real del FPS, ya que en caso contrario, no se garantiza que el ∂FPS sea el contorno real del FPS (ver figura 7.1).

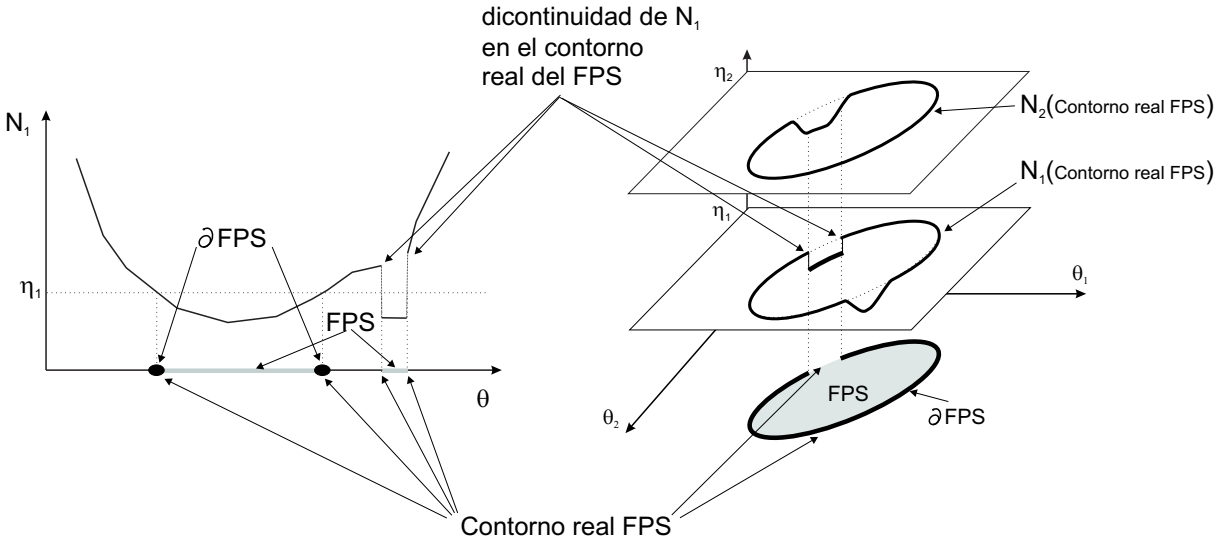

Figura 7.1: Situación donde el ∂FPS no representa el contorno real del FPS debido a la discontinuidad de $N_1(\theta)$ en el contorno real del FPS. A la izquierda caso $s = 1$ y $L = 1$ y a la derecha $s = 2$ y $L = 2$ ($\theta \in \mathcal{R}^L$).

B) Otra alternativa es caracterizar el FPS y el ∂FPS al mismo tiempo. Para ello se plantea una función $J'(\theta)$ de manera que sus mínimos globales constituyan el ∂FPS y donde el FPS constituya soluciones globales quasi mínimas. Esta solución obtiene el ∂FPS y el FPS directamente aún cuando $N_i(\theta)$ no sea continua en el contorno real del FPS.

Caso A)

La función $J(\theta)$ que se plantea para obtener el ∂FPS se formula de la siguiente manera:

$$J(\theta) := \begin{cases} \sum_B J_i(\theta) & si \quad B(\theta) \neq \emptyset \\ \\ \prod_A J_i(\theta) & si \quad B(\theta) = \emptyset \end{cases}, \tag{7.6}$$

donde:

$$B(\theta) : = \{i \in A : N_i(\theta) > \eta_i\}, \tag{7.7}$$

$$J_i(\theta) = |N_i(\theta) - \eta_i|. \tag{7.8}$$

Algunas de las propiedades de la función $J(\theta)$ son:

1. Semidefinida positiva, ya que J_i también lo es.

2. $B(\theta) = \emptyset$ cuando $\theta \in FPS$, $J_i(\theta) = 0$ si $\theta \in \partial FPS_i$ y $J(\theta) = 0$ si $\theta \in \partial FPS$.

3. Continua si $N_i(\theta)$ es continua $\forall i \in A$.

   Demostración. *Si $N_i(\theta)$ es continua también lo será $J_i(\theta)$. Dado que $J(\theta)$ es una función a tramos, se ha de demostrar su continuidad en los diferentes tramos y en la unión de éstos:*

   - *Los diferentes tramos se componen de la suma o la multiplicación de $J_i(\theta)$, por lo tanto, dado que la suma y multiplicación de funciones continuas es otra función continua, se asegura la continuidad dentro de dichos tramos.*
   - *La unión entre tramos se produce por la aparición o desaparición de componentes $J_i(\theta)$ en el sumatorio o al pasar del sumatorio al productorio o viceversa. Sin pérdida de generalidad se asume que $s = 2$, por lo tanto una unión entre tramos se puede dar al pasar, en cualquiera de los dos sentidos de:*

$$\begin{array}{lll} (a) & J(\theta) = J_1(\theta) & \leftrightarrows J(\theta) = J_1(\theta) + J_2(\theta), \\ (b) & J(\theta) = J_2(\theta) & \leftrightarrows J(\theta) = J_1(\theta) + J_2(\theta), \\ (c) & J(\theta) = J_1(\theta) & \leftrightarrows J(\theta) = J_1(\theta)J_2(\theta), \\ (d) & J(\theta) = J_2(\theta) & \leftrightarrows J(\theta) = J_1(\theta)J_2(\theta), \\ (e) & J(\theta) = J_1(\theta) + J_2(\theta) & \leftrightarrows J(\theta) = J_1(\theta)J_2(\theta). \end{array}$$

   El caso (a) se dará cuando $B(\theta)$ pase de $B(\theta) = \{1\}$ debido a que $N_1(\theta) > \eta_1$ y $N_2(\theta) \leq \eta_2$ a $B(\theta) = \{1, 2\}$, debido a que $N_2(\theta) > \eta_2$ y por lo tanto la discontinuidad hay que estudiarla en el punto θ' para el cual $N_2(\theta') = \eta_2$ y $J_2(\theta') = 0$. De modo que

$$J(\theta') = \lim_{\theta \to \theta'^+} J(\theta) = J_1(\theta') = \lim_{\theta \to \theta'^-} J(\theta) = J_1(\theta') + J_2(\theta').$$

   El caso (c) se dará cuando $B(\theta)$ pase de $B(\theta) = \{1\}$ debido a que $N_1(\theta) > \eta_1$ y $N_2(\theta) \leq \eta_2$ a $B(\theta) = \emptyset$, debido a que $N_1(\theta) \leq \eta_1$ y por lo tanto la discontinuidad hay que estudiarla en el punto θ' para el cual $N_1(\theta') = \eta_1$ y $J_1(\theta') = 0$. De modo que

$$J(\theta') = \lim_{\theta \to \theta'^+} J(\theta) = J_1(\theta') = \lim_{\theta \to \theta'^-} J(\theta) = J_1(\theta')J_2(\theta') = 0.$$

   siendo θ' en este caso un mínimo global de $J(\theta)$.

   Los casos (b) y (d) son idénticos al (a) y (c) sustituyendo $J_1(\theta)$ por $J_2(\theta)$ respectivamente y el caso (e) se obtiene por la aplicación simultánea de los casos (c) y (d). Con lo cual se demuestra la continuidad de $J(\theta)$.

$\Diamond$

A continuación se procede a analizar, de forma gráfica, como se constituye $J(\theta)$ para un caso donde $s = 2$ (se utilizan dos normas) y $L = 1$ (es decir $\theta \in \mathcal{R}$)[3].

$$\eta_1 = 20 \Rightarrow J_1(\theta) = \left|N_1(\theta) - 20\right|,$$
$$\eta_2 = 35 \Rightarrow J_2(\theta) = \left|N_2(\theta) - 35\right|.$$

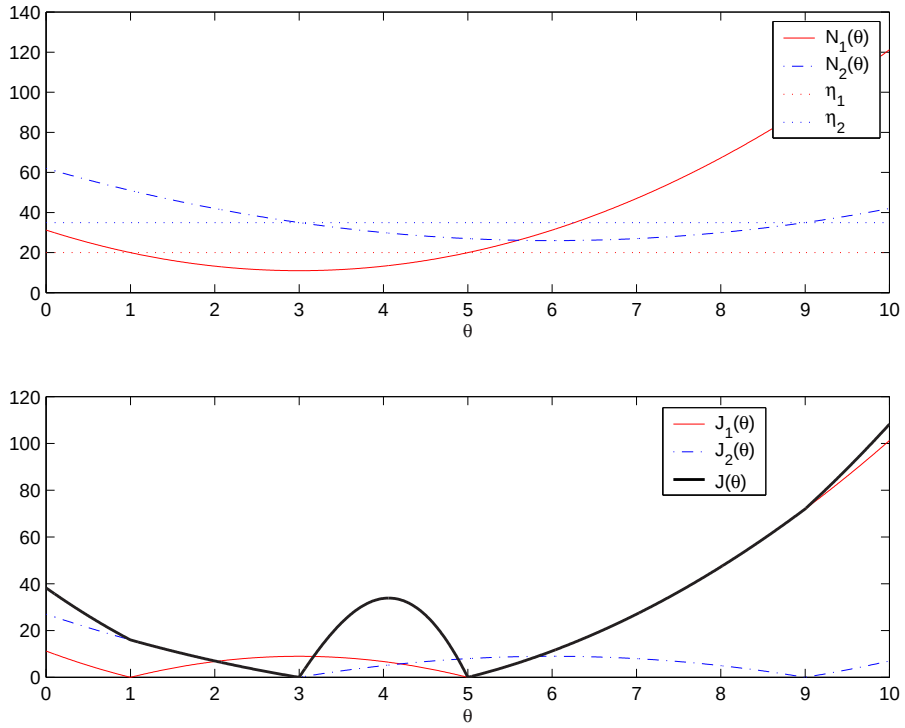

Figura 7.2: Ilustración de la construcción de la función $J(\theta)$ para un caso donde $s = 2$ y $L = 1$. Arriba se representan las dos normas $N_1(\theta)$ y $N_2(\theta)$ y sus respectivas cotas $\eta_1 = 20$ y $\eta_2 = 25$. Bajo $J_1(\theta)$, $J_2(\theta)$ y la función resultante $J(\theta)$.

La figura 7.2 ilustra la composición de $J(\theta)$ y pone de manifiesto su característica de semidefinida positiva y continua, ya que, $N_1(\theta)$ y $N_2(\theta)$ también lo son. Por otra parte se observa que

$$FPS_1 := \{\theta : \theta \in [1 \ldots 5]\},$$
$$FPS_2 := \{\theta : \theta \in [3 \ldots 9]\},$$

y por lo tanto, como se puede observar

$$FPS := \{\theta : \theta \in [3 \ldots 5]\},$$

$$\partial FPS = \{3, 5\}, \quad J(3) = J(5) = 0.$$

[3]Para que la notación escogida no dé pie a confusión se remarca que $N_1(\theta)$ y $N_2(\theta)$ representan dos normas cualesquiera del error de identificación, por ejemplo, N_1 la norma-∞ y N_2 la norma-1.

Para este caso particular, propiciado porque $L = 1$, el ∂FPS es un conjunto finito que contiene sólo dos puntos. Sin embargo, lo normal, si $FPS \neq \emptyset$, es que el ∂FPS sea un conjunto infinito de soluciones mínimas globales. Para caracterizar el ∂FPS se utilizará el algoritmo ϵ-GA que generará un conjunto finito de soluciones ∂FPS^* distribuido a lo largo del ∂FPS que sirva para caracterizarlo.

A partir del ∂FPS^* podría caracterizarse el FPS tomando puntos en su interior (por ejemplo, utilizando una distribución uniforme mediante una rejilla) sabiendo que los puntos $\theta \in FPS$, deben cumplir $N_i(\theta) \leq \eta_i,\ \forall i \in A$.

Esta alternativa, que correspondería a la primera de las comentadas anteriormente, no podría garantizar la caracterización del FPS cuando $N_i(\theta)$ es discontinua en el contorno real del FPS[4].

Caso B)

En esta segunda alternativa se caracteriza el ∂FPS y el FPS al mismo tiempo. Para ello se optimiza la siguiente función $J'(\theta)$

$$J'(\theta) := \begin{cases} \sum_B J_i & si \quad B(\theta) \neq \emptyset \\ \\ \text{mín}(\delta, \prod_A J_i) & si \quad B(\theta) = \emptyset \end{cases} \tag{7.9}$$

muy similar a la función $J(\theta)$ anterior y con sus mismas propiedades (ver figura 7.3). Además, dado que el mínimo global de $J'(\theta)$ es $J^* = 0$ (cuando $\theta \in \partial FPS$) y como $J'(\theta) < \delta$ cuando $\theta \in FPS$ ($B(\theta) = \emptyset$) se asegura que estas soluciones sean soluciones globales quasi mínimas y por lo tanto nunca serán eliminadas del archivo $A(t)$ al utilizar el algoritmo ϵ-GA[5]. De este modo el algoritmo generará una caracterización del FPS, el conjunto finito FPS^*. Al seguir manteniendo los mínimos globales en el ∂FPS se fomenta su caracterización frente a los individuos que pertenecen al FPS y no al ∂FPS.

En cualquiera de las dos alternativas barajadas, el FPS vendrá condicionado por los diferentes FPS_i y éstos a su vez por sus normas N_i y sus correspondientes cotas η_i. A la hora de establecer las cotas a aplicar sobre N_i se debe utilizar la información que se tenga *a priori* del proceso (p.e. dinámica no modelada) y las características del ruido. Sin embargo, esto puede resultar complicado y fomenta que, en muchos casos, se terminen eligiendo las cotas atendiendo a las prestaciones que se desean para las predicciones del modelo. Una elección inadecuada podría resultar en un FPS conservativo, si se escogen cotas demasiado grandes o en un $FPS = \emptyset$ si las cotas son muy bajas.

En el caso de utilizar una única norma $N_1(\theta)$, resulta interesante como manifiestan otros autores [160], para evitar esta última situación ($FPS = \emptyset$), determinar la cota correspondiente mediante la minimización de la norma $N_1(\theta)$ que darían la cota inferior $\eta_1^{min} = \text{mín}_\theta\, N_1(\theta)$ y se garantizaría un $FPS \neq \emptyset$ escogiendo $\eta_1 \geq \eta_1^{min}$.

Cuando se utilizan varias normas simultáneamente, se pueden obtener las cotas mínimas $\eta_i^{min} = \text{mín}_\theta\, N_i(\theta)$ pero, en este caso el escoger $\eta_i \geq \eta_i^{min}$ no implica que $FPS \neq \emptyset$

[4]Situación que podría venir producida por alguna no linealidad del modelo.

[5]Ver la definición 5.2 de solución mínima global y su relación con δ en el capítulo 5.

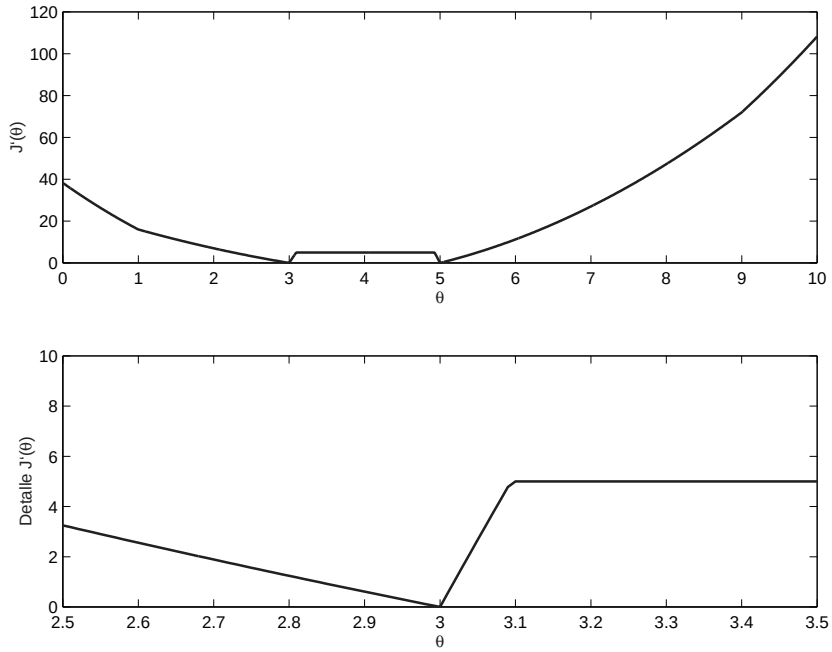

Figura 7.3: Arriba la función $J'(\theta)$ para el mismo caso ilustrado en la figura 7.3. Debajo detalle de una zona donde se encuentra un mínimo local en el ∂FPS. En este caso se ha utilizado un $\delta = 5$.

como se demostrará posteriormente. La alternativa que se propone en esta tesis para escoger las cotas η_i pasa por realizar la optimización de las normas N_i de forma simultánea, a través de una optimización multiobjetivo[6]. Por lo tanto, el problema de optimización multiobjetivo a resolver sería el siguiente:

$$\min_{\theta \in D} \mathbf{J}(\theta) \tag{7.10}$$

donde

$$\mathbf{J}(\theta) = \{N_1(\theta), N_2(\theta), \ldots, N_s(\theta)\}.$$

La solución al problema de optimización será el frente de Pareto que contendrá las estimaciones $\hat{\mathbf{\Theta}}_P$ o modelos óptimos de proyección con respecto a las diferentes normas al mismo tiempo.

Definición 7.5 (Modelos óptimos de proyección $\hat{\mathbf{\Theta}}_P$): *Se define $\hat{\mathbf{\Theta}}_P$, el conjunto de modelos óptimos de proyección, como las soluciones de Pareto que se obtienen al resolver el problema de optimización multiobjetivo definido en (7.10):*

$$\hat{\mathbf{\Theta}}_P = \{\theta \in D \mid \nexists \tilde{\theta} \in D \ : \ \mathbf{J}(\tilde{\theta}) \preceq \mathbf{J}(\theta)\} \ . \tag{7.11}$$

$\triangle$

[6]Al realizar una optimización multiobjetivo se reduce el coste computacional, ya que sólo se realiza una optimización, y no s optimizaciones independientes de las N_i respectivas ($i \in s$).

Para determinar $\hat{\mathbf{\Theta}}_P$ se utilizará el algoritmo $\not\epsilon$-MOGA que obtendrá como solución un conjunto finito de modelos óptimos distribuidos a lo largo del frente de Pareto, $\hat{\mathbf{\Theta}}_P^*$. Dado que $\hat{\mathbf{\Theta}}_P^*$ captura los extremos del frente de Pareto en $\hat{\mathbf{\Theta}}_P^*$ estarán las estimaciones de proyección independientes $\hat{\theta}_{N_i}$ de cada N_i planteada.

$$\hat{\theta}_{N_i} = \arg \min_{\theta \in FPS} N_i. \tag{7.12}$$

Una vez resuelto el problema de optimización se puede utilizar la información que genera el frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P^*)$ en su conjunto para elegir las cotas η_i como a continuación se muestra. La figura 7.4 ilustra un caso en el que se utilizan dos normas N_1 y N_2 del error de identificación. Sobre el frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P)$ que aparece en la figura se ha resaltado un fragmento de éste que depende de las cotas η_1 y η_2 escogidas y corresponde a los modelos óptimos de proyección restringida $\mathbf{J}(\hat{\mathbf{\Theta}}_{Pr})$.

Definición 7.6 (Modelos óptimos de proyección restringida $\hat{\mathbf{\Theta}}_{Pr}$): *Se define $\hat{\mathbf{\Theta}}_{Pr}$, el conjunto de modelos óptimos de proyección restringida, como el conjunto de modelos óptimos de proyección del FPS:*

$$\hat{\mathbf{\Theta}}_{Pr} = \hat{\mathbf{\Theta}}_P \bigcap FPS. \tag{7.13}$$

△

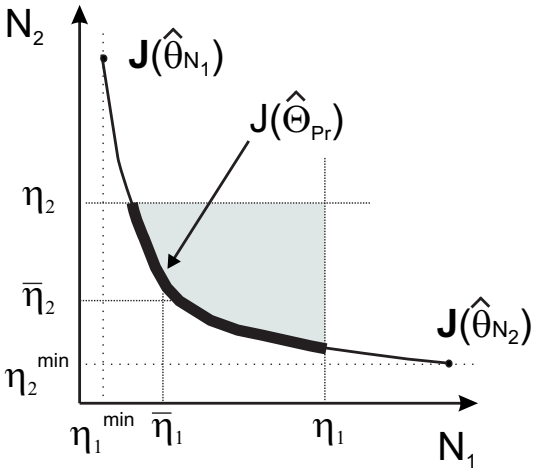

Figura 7.4: Frente de Pareto para un caso donde $s = 2$. Representación de las cotas mínimas η_1^{min} y η_2^{min} así como de la imagen, sobre el frente de Pareto, de los modelos de proyección restringida $\hat{\Theta}_{Pr}$ en función de las cotas $\eta_1 > \eta_1^{min}$ y $\eta_2 > \eta_2^{min}$ escogidas. Las cotas $\bar{\eta}_1 > \eta_1^{min}$ y $\bar{\eta}_2 > \eta_2^{min}$, sin embargo, generarían un $FPS = \emptyset$.

Por lo tanto, para asegurar que $FPS \neq \emptyset$, ya que $\hat{\mathbf{\Theta}}_{Pr} \subset FPS$, es suficiente con escoger η_i de manera que $\hat{\mathbf{\Theta}}_{Pr} \neq \emptyset$. En el ejemplo representado en la figura se muestra como con las cotas η_1 y η_2 (superiores a las cotas mínimas η_1^{min} y η_2^{min} respectivamente) escogidas se consigue que $\hat{\mathbf{\Theta}}_{Pr} \neq \emptyset$. La zona que aparece sombreada englobaría la imagen

del FPS sobre el espacio (N_1, N_2), es decir, $\mathbf{J}(FPS)$ pertenecería a la zona sombreada que queda delimitada por las cotas η_1, η_2 y el propio frente de Pareto. Si se hubiesen utilizado las cotas $\bar{\eta}_1$ y $\bar{\eta}_2$ (también superiores a las cotas mínimas η_1^{min} y η_2^{min} respectivamente) $\hat{\mathbf{\Theta}}_{Pr} = \emptyset$ y por lo tanto $FPS = \emptyset$ (no existiría zona sombreada) demostrándose así que no es suficiente con que $\bar{\eta}_i > \eta_i^{min}$ para que $FPS \neq \emptyset$.

Con lo cual la obtención de $\mathbf{J}(\hat{\mathbf{\Theta}}_P)$ servirá:

- Como paso intermedio en la obtención del FPS para decidir los valores de η_i que aseguren un $FPS \neq \emptyset$.
- Para determinar los modelos óptimos de proyección restringida $\hat{\mathbf{\Theta}}_{Pr} \subset FPS$ candidatos a ser modelo nominal.
- Como ayuda para escoger el espacio de búsqueda a utilizar en el proceso de obtención del FPS en función de la ubicación de $\hat{\mathbf{\Theta}}_P$.
- Para ver la interacción entre las diferentes normas (en función de la extensión del frente) y mostrar aspectos relacionados con las prestaciones del modelo (p.e. en función de las cotas mínimas obtenidas).

Una vez escogidas las cotas η_i y caracterizado el FPS mediante el FPS^* y/o el ∂FPS^* se puede aproximar el modelo nominal óptimo de peor caso (como un posible modelo nominal) calculando el centro de Chebyshev del FPS^* (o del ∂FPS cuando $N_i(\theta)$ son continuas) mediante:

$$\hat{\theta}_c^* = arg \min_{\theta \in \mathbf{D}} \max_{\overline{\theta} \in \mathbf{FPS}^*} \|\theta - \overline{\theta}\|_2, \tag{7.14}$$

ó

$$\hat{\theta}_c^* = arg \min_{\theta \in \mathbf{D}} \max_{\overline{\theta} \in \partial \mathbf{FPS}^*} \|\theta - \overline{\theta}\|_2. \tag{7.15}$$

Dado que el FPS^* y el ∂FPS^* son conjuntos finitos, la parte correspondiente a la maximización se puede determinar fácilmente como:

$$F(\theta) = \max_{\overline{\theta} \in \mathbf{FPS}^*} \|\theta - \overline{\theta}\|_2 \tag{7.16}$$

ó

$$F(\theta) = \max_{\overline{\theta} \in \partial \mathbf{FPS}^*} \|\theta - \overline{\theta}\|_2 \tag{7.17}$$

y por lo tanto el problema de minimización se plantearía de la siguiente manera:

$$\hat{\theta}_c^* = arg \min_{\theta \in \mathbf{D}} F(\theta). \tag{7.18}$$

Otra alternativa que se puede utilizar para determinar un posible modelo nominal óptimo, en términos del error de identificación y, al mismo tiempo en términos del error

de estimación en el espacio de los parámetros, consistiría en obtener el modelo óptimo de proyección restringida que más cerca estuviese del centro del FPS.

Definición 7.7 (Modelo óptimo de proyección interpolada restringida $\hat{\theta}_{pi}$): *Se define $\hat{\theta}_{pi}$, el modelo óptimo de proyección interpolada restringida, como el modelo óptimo de proyección restringida más cercano al centro del FPS:*

$$\hat{\theta}_{pi} = \min_{\theta \in \hat{\mathbf{\Theta}}_{Pr}} \|\theta - \hat{\theta}_c\|, \tag{7.19}$$

por lo tanto, dado que $\hat{\mathbf{\Theta}}_{Pr} \subset FPS$, $\hat{\theta}_{pi} \subset FPS$.

△

Si se dispone de $\hat{\mathbf{\Theta}}^*_{Pr}$ y de $\hat{\theta}^*_c$ se puede obtener una aproximación a $\hat{\theta}_{pi}$ de forma más sencilla.

$$\hat{\theta}^*_{pi} = \min_{\theta \in \hat{\mathbf{\Theta}}^*_{Pr}} \|\theta - \hat{\theta}^*_c\|. \tag{7.20}$$

7.2.1. Validación del FPS

Una vez determinado el conjunto de parámetros factibles a través del proceso de identificación robusta FPS_{ide}, utilizando los datos de un determinado experimento $\Omega_{ide} = \{\mathbf{Y}_{ide}, \mathbf{U}_{ide}\}$, las s normas N_i y sus correspondientes cotas η_i, debe validarse utilizando otros datos diferentes $\Omega_{val} = \{\mathbf{Y}_{val}, \mathbf{U}_{val}\}$. Para ello, es necesario establecer un criterio de validación adecuado.

Una posibilidad es validar el FPS_{ide} comprobando si contiene modelos consistentes con los nuevos datos Ω_{val}. Esto sería equivalente a decir que el FPS que se obtendría del proceso de identificación con los datos $\Omega = \{\Omega_{ide}, \Omega_{val}\}$ sería un $FPS \neq \emptyset$. La figura 7.5 muestra gráficamente dos situaciones, una donde hay modelos en el FPS_{ide} que al mismo tiempo también pertenezca al FPS_{val} (conjunto de parámetros consistente con Ω_{val} y las mismas s normas N_i y cotas η_i utilizadas en la identificación del FPS_{ide}) por lo tanto, el FPS_{ide} quedaría validado y otra donde no se da esta situación y por lo tanto el FPS_{ide} quedaría invalidado.

Si el FPS_{ide} resulta validado, el FPS resultante sería $FPS = FPS_{ide} \bigcap FPS_{val}$. No es necesario obtener el FPS_{val}, simplemente dejar en el FPS los modelos del FPS_{ide} que son consistentes con los datos Ω_{val}. Dado que se dispone del FPS^*_{ide}, que es un conjunto finito, la obtención del FPS^* es muy sencilla pues sólo hay que simular los modelos $\theta \in FPS^*_{ide}$ (utilizando Ω_{val}) y quedarse con aquellos que cumplen $N_i(\theta) \leq \eta_i \; \forall i \in A$.

Si el FPS_{ide} no es validado se podría actuar de varias formas:

- Aumentar las cotas η_i o algunas de ellas hasta conseguir un FPS_{ide} que pudiese ser validado con los datos Ω_{val}.
- Modificar el modelo (por ejemplo, añadiendo parte de la dinámica no modelada) hasta que el FPS_{ide} quedase validado.

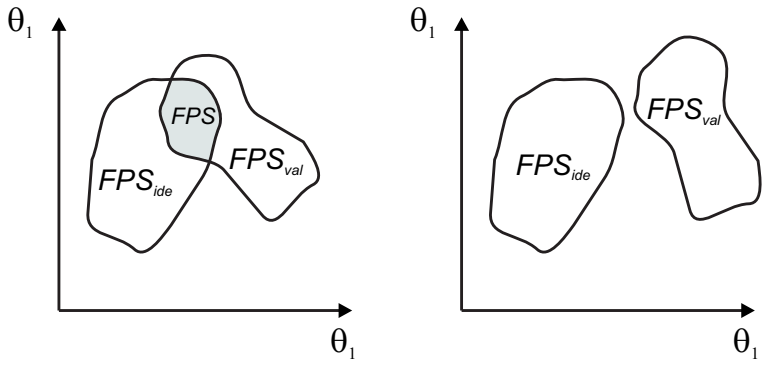

Figura 7.5: Proceso de validación. A la izquierda el FPS_{ide} queda validado, ya que $FPS \neq \emptyset$. A la derecha en cambio el FPS_{ide} no es validado debido a que $FPS = \emptyset$.

En esta segunda opción no se tendrían que aumentar las cotas η_i y por lo tanto no se perderían prestaciones en las predicciones del modelo como ocurría en la primera opción, sin embargo, el modelo sería, seguramente, más complejo.

Del mismo modo que se puede validar el FPS_{ide}, se puede validar un determinado modelo óptimo $\hat{\theta} \in FPS$ simplemente comprobando que $\hat{\theta} \in FPS_{val}$ o lo que es lo mismo simular (utilizando Ω_{val}) el modelo $\hat{\theta}$ y comprobar que cumple que $N_i(\hat{\theta}) \leq \eta_i \; \forall i \in A$.

Otra posibilidad, que plantean algunos autores [118], para validar el FPS_{ide} es que las salidas producidas por los modelos del FPS_{ide} envolviesen los datos Ω_{val}.

Definición 7.8 (Envolvente $\hat{\mathcal{Y}}(\Theta, \mathbf{U})$): *Se define $\hat{\mathcal{Y}}(\Theta, U)$, como la envolvente de las respuestas de los modelos $\theta \in \Theta$ cuando se aplica la entrada $\mathbf{U}$:*

$$\hat{\mathcal{Y}}(\Theta, \mathbf{U}) = \{\hat{\mathbf{Y}}^{min}, \hat{\mathbf{Y}}^{max}\}, \quad \hat{\mathbf{Y}}^{min}, \hat{\mathbf{Y}}^{max} \subseteq \mathcal{R}^{l \times N}, \tag{7.21}$$

donde los elementos "ji" de $\hat{\mathbf{Y}}^{min}$ e $\hat{\mathbf{Y}}^{max}$ se determinan como:

$$\hat{y}_j^{min}(t_i) = \min_{\theta \in \Theta} \hat{y}_j^{min}(t_i, \theta)$$

$$\hat{y}_j^{max}(t_i) = \max_{\theta \in \Theta} \hat{y}_j^{min}(t_i, \theta).$$

$\triangle$

Por lo tanto el FPS_{ide} quedaría validado si $\hat{\mathcal{Y}}(FPS_{ide}, \mathbf{U}_{val})$ envuelve también a $\mathbf{Y}_{val}$, es decir,

$$\hat{y}_j^{min}(t_i) \leq y_j^{val}(t_i) \leq \hat{y}_j^{max}(t_i), \quad \forall j \in [1 \ldots l], i \in [1 \ldots N].$$

Al utilizar este criterio de validación, podría darse la situación en la que dado un FPS_{ide} la envolvente $\hat{\mathcal{Y}}(FPS_{ide}, \mathbf{U}_{ide})$ no envolviese $\mathbf{Y}_{ide}$, es decir, no se podría validar el FPS_{ide} ni siquiera para los datos Ω_{ide}.

Esta situación se debe a que el procedimiento de obtención del FPS_{ide} difiere del criterio de validación y se manifiesta de forma especial cuando existen errores de modelado

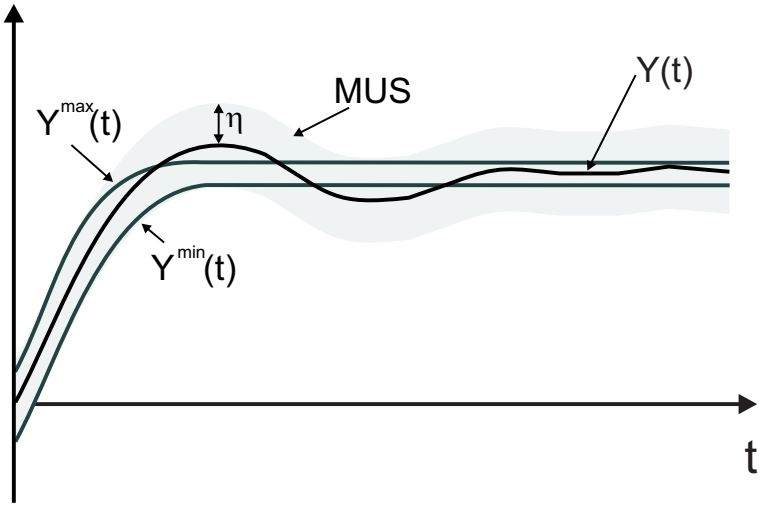

Figura 7.6: Proceso de IR donde la envolvente envolvente $\{\hat{\mathbf{Y}}^{min}, \hat{\mathbf{Y}}^{max}\}$ no envuelve la respuesta del proceso $\mathbf{Y}(t)$.

evidentes. La figura 7.6 muestra un ejemplo donde se modela una respuesta subamortiguada mediante un modelo de primer orden y se utiliza una única norma $N(\theta) = ||e(\theta)||_\infty$. En la figura se muestra el MUS[7] asociado a dicha norma así como la envolvente del FPS_{ide} que se obtendría en el proceso de IR. Como se puede observar las envolventes $\mathbf{Y}^{max}(t)$ y $\mathbf{Y}^{min}(t)$ no envuelven la propia respuesta $\mathbf{Y}(t)$ utilizada en el proceso de IR. Para conseguirlo habría que aumentar considerablemente la cota η generando un FPS_{ide} mayor.

Aunque este criterio de validación puede resultar demasiado conservativo, el uso de la envolvente puede ayudar a detectar posibles errores de modelado o a valorar si se está siendo muy conservativo con las cotas elegidas.

7.2.2. Resumen

A continuación, se muestran a modo de resumen los pasos a seguir en el proceso de IR propuesto:

1. Determinar y procesar, si procede, los datos que se utilizarán en el proceso de identificación Ω_{ide} y validación Ω_{val}.

2. Establecer una estructura inicial para el modelo con los parámetros a identificar.

3. Plantear el número y tipo de normas a utilizar.

4. Utilizar el algoritmo $\not\epsilon$-MOGA para determinar los modelos óptimos de proyección.

[7]Conjunto de incertidumbre en las salidas (ver la definición 6.5 del capítulo 6).

5. A partir de dichos modelos determinar los parámetros que se someterán al proceso de IR y las cotas η_i que caracterizarán el FPS_{ide}.

6. Utilizar el algoritmo ϵ-GA para optimizar la función $J(\theta)$ o $J'(\theta)$ que generará el ∂FPS^*_{ide} o el FPS^*_{ide}.

7. A partir del FPS^*_{ide}, si se desea, determinar el modelo de peor caso, como una aproximación al centro de Chebyshev del FPS^*_{ide} correspondiente, es decir, $\hat{\theta}^*_c$.

8. Con $\hat{\theta}^*_c$ y los modelos de proyección restringida $\hat{\Theta}^*_{Pr}$ determinar el modelo óptimo de proyección interpolada restringida $\hat{\theta}^*_{pi}$.

9. Validar el FPS^*_{ide} y los modelos nominales, si procede, utilizando Ω_{val}. Si el proceso de validación no se supera satisfactoriamente repetir el proceso modificando, las cotas y/o las normas y/o el modelo.

7.3. Ejemplo 1. Sistema Térmico

El objetivo de esta sección es ilustrar, a través de la identificación robusta de una maqueta de un horno (figura 7.7) en la cual se estudian los comportamientos típicos de los procesos térmicos, la metodología planteada en la sección anterior. El aporte de energía en el interior del proceso se debe a la potencia disipada por la resistencia que se encuentra en su interior. La circulación del aire en el interior del proceso se produce gracias a un ventilador, que introduce continuamente y de forma constante aire del exterior.

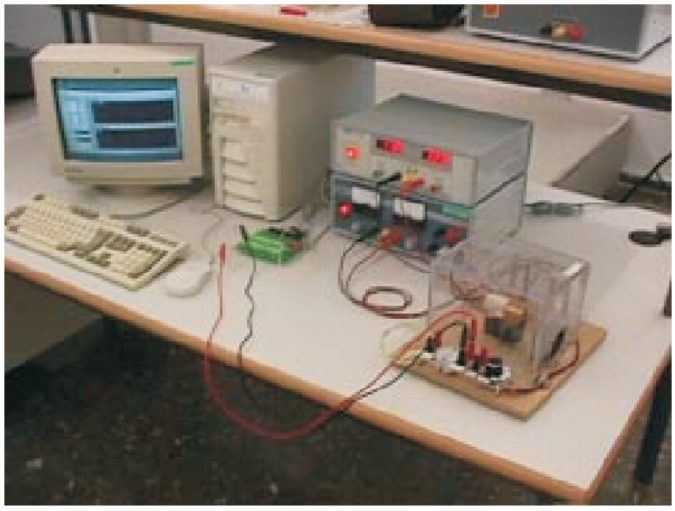

Figura 7.7: Imagen del Proceso Térmico.

El actuador lo constituye una fuente de tensión controlada por tensión, el rango de entrada del actuador es $0 \div 100\,\% \Rightarrow 0 \div 7.5$v. Se utilizan dos termopares para medir la temperatura de la resistencia y la del aire en el rango $-50 \div 250^{o}C$.

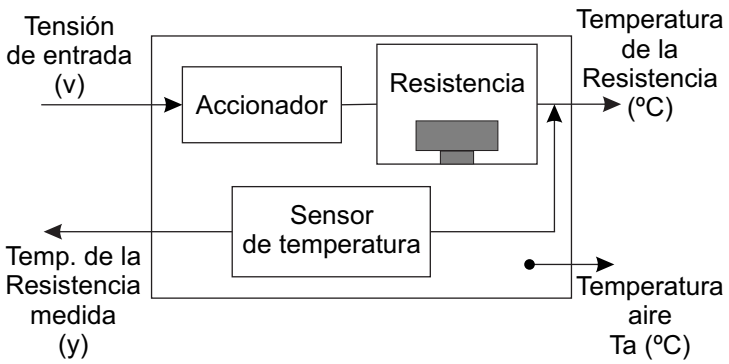

Figura 7.8: Esquema del prototipo: Proceso Térmico.

La figura 7.9 muestra las señales que se utilizarán en el proceso de identificación, mientras que en la figura 7.10 se muestran las que se utilizarán para la validación.

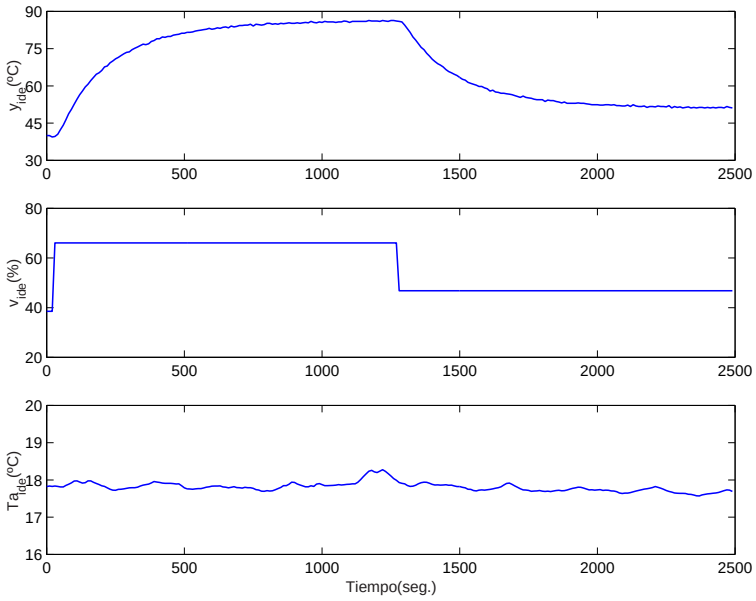

Figura 7.9: Datos $\Omega_{ide} = \{y_{ide}(t), U_{ide}(t)]\}$ con $U_{ide}(t_i) = [v_{ide}(t_i), Ta_{ide}(t_i)]^T$ para la identificación. $y_{ide}(t)$ temperatura de salida del proceso, $v_{ide}(t)$ entrada del proceso y $Ta_{ide}(t)$ la perturbación producida por la temperatura del aire. $N = 250$ (longitud del experimento) con $T = 10seg.$ (periodo de muestreo).

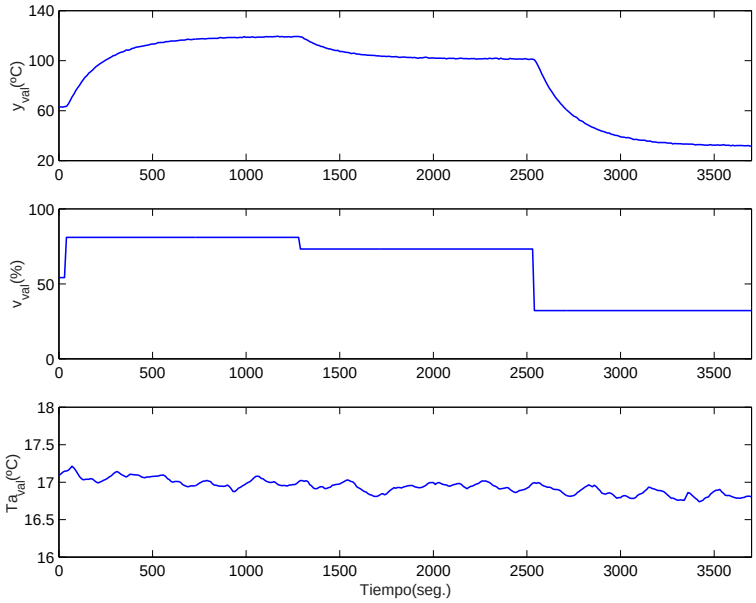

Figura 7.10: Datos $\Omega_{val} = \{y_{val}(t), U_{val}(t)]\}$ con $U_{val}(t_i) = [v_{val}(t_i), Ta_{val}(t_i)]^T$ para la validación. $y_{val}(t)$ temperatura de salida del proceso, $v_{val}(t)$ entrada del proceso y $Ta_{val}(t)$ la perturbación producida por la temperatura del aire. $N = 371$ y $T = 10seg.$

7.3.1. Modelo con dos parámetros

Atendiendo a las leyes físicas de la termodinámica [166] se puede definir la estructura inicial del modelo mediante las siguientes ecuaciones diferenciales, donde se modelan las pérdidas de calor por convección y conducción:

$$\begin{aligned} \dot{x}(t) &= \frac{1}{1000}\left(\theta_1 v(t)^2 - \theta_2\left(x(t) - Ta(t)\right)\right), \\ \hat{y}(t) &= x(t), \end{aligned} \tag{7.22}$$

donde:

- $\dot{x}(t)$ es la variable de estado del modelo,
- $v(t)$ es la tensión de entrada al proceso en voltios $0 \div 100(\,\%) \Rightarrow 0 \div 7.5$v,
- $\hat{y}(t)$ es la temperatura de salida de proceso en (^oC),
- $Ta(t)$ es la temperatura del aire en (^oC) y
- $\theta = [\theta_1, \theta_2]^T$ son los parámetros del modelo.

Para simular el modelo es necesario conocer el valor del estado en el instante inicial. En este caso el estado es medible y se podría inicializar $x(0) = y(0)$. Esto podría condicionar

el resultado final, si se tiene en cuenta que $y(0)$ está afectada por el ruido de medida y que existen errores de modelado. Se puede evitar este problema intentando independizar, en la medida de lo posible, el resultado de la estimación de los parámetros de las condiciones iniciales que se tomen. Para ello se fijan los estados a partir de observaciones de las salidas y no se tiene en cuenta el error de identificación producido durante las φ muestras iniciales. La determinación de φ es importante, ya que un φ muy elevado provocaría una pérdida excesiva de datos iniciales, pues no tendrían repercusión sobre la estimación y un φ muy pequeño desvirtuaría el procedimiento.

Otras alternativas para inicializar el estado podrían ser las siguientes:

- Añadirlo como un parámetro más a identificar $\theta = [\theta_1, \theta_2, x(0)]^T$. Esta alternativa incrementa la dimensión del espacio de búsqueda aumentando el tiempo y la dificultad del proceso de optimización. Por otra parte la condición inicial que se obtuviese no serviría para ser utiliza como condición inicial para otros datos diferentes.
- Asumir, por ejemplo, que el sistema se encuentra en régimen permanente[8] y por lo tanto, $\dot{x}(0) = 0$. De esta manera se podría determinar $x(0)$ despejando de la ecuación del modelo:
$$x(0) = \frac{1}{\theta_2}\left(\theta_1 v(0)^2 + \theta_2 T a(0)\right).$$
Así se evita tener que identificar $x(0)$ aunque se puede cometer error al asumir que $\dot{x}(0) = 0$. Por otra parte la obtención de $x(0)$ de forma explícita no siempre es posible o podrían existir varias soluciones de las cuales habría que seleccionar una de ellas.

Para este ejemplo en concreto se va a inicializar $x(0) = y(0)$ y se fijará $\varphi = 20$ muestras. Se ha escogido un φ pequeño ya que el sistema, al inicio del experimento, se encuentra prácticamente en régimen permanente.

Para determinar el FPS_{ide} asociado a los parámetros θ se van a plantear dos normas sobre el error de identificación. La norma infinito (N_1) y la norma absoluta (N_2) asumiendo así que el error de identificación estará acotado en cada medida tomada sobre la salida y en valor medio sobre su módulo a lo largo del experimento (integral del módulo del error de identificación).

$$N_1(\theta) = \|\mathbf{e}(\theta)\|_\infty^{\mathbf{w}}, \;\; \mathbf{w}_i = 0 \; \forall i \in [1, \ldots, \varphi], \; \mathbf{w}_i = 1 \; \forall i \in [\varphi + 1, \ldots, N], \tag{7.23}$$

$$N_2(\theta) = \|\mathbf{e}(\theta)\|_1^{\mathbf{w}}, \;\; \mathbf{w}_i = 0 \; \forall i \in [1, \ldots, \varphi], \; \mathbf{w}_i = \frac{1}{(N - \varphi)} \; \forall i \in [\varphi + 1, \ldots, N]. \tag{7.24}$$

Por lo tanto la función (7.9) a optimizar se formularía a partir de:

$$J_1(\theta) = \left|N_1(\theta) - \eta_1\right|, \tag{7.25}$$

[8] Como se puede observar en la figura 7.9 la salida se encuentra prácticamente en régimen permanente.

$$J_2(\theta) = \left|N_2(\theta) - \eta_2\right|. \tag{7.26}$$

Tal y como se planteó en la sección anterior, para escoger las cotas η_1 y η_2, se utilizará la información que genere el frente de Pareto del siguiente problema de optimización multiobjetivo.

$$\min_{\theta \in D} \mathbf{J}(\theta) = \{N_1, N_2\}. \tag{7.27}$$

Los parámetros del algoritmo ~~ε~~MOGA escogidos son los siguientes[9]:

- El espacio de búsqueda[10] D es $\theta_1 \in [0.01 \ldots 0.3]\frac{^oC}{seg.\,\%^2}$, $\theta_2 \in [2.0 \ldots 20.0]\frac{1}{seg.}$.
- $t_{max} = 500$ y $n\_box_1 = n\_box_2 = 50$ para que el *grid* del espacio de las funciones objetivo tenga 50 divisiones en cada dimensión.

La figura 7.11 muestra los modelos óptimos de proyección $\hat{\mathbf{\Theta}}_P^*$ obtenidos como solución al problema de optimización multiobjetivo y el frente de Pareto originado.

Del análisis del frente de Pareto se puede obtener la siguiente información útil para la determinación del FPS_{ide}:

- El frente es pequeño lo que indica que ambas normas están estrechamente relacionadas y los modelos de proyección están bastante cerca unos de otros.
- Los modelos de proyección $\hat{\theta}_{N_1}$ y $\hat{\theta}_{N_2}$ son los siguientes:

$$\hat{\theta}_{N_1} = [0.0768, 4.898]^T \Rightarrow N_1(\hat{\theta}_{N_1}) = \mathbf{1.260}^oC,\ N_2(\hat{\theta}_{N_1}) = 0.475^oC,$$
$$\hat{\theta}_{N_2} = [0.0780, 5.010]^T \Rightarrow N_1(\hat{\theta}_{N_2}) = 1.802^oC,\ N_2(\hat{\theta}_{N_2}) = \mathbf{0.303}^oC.$$

Teniendo en cuenta que el modelo es muy sencillo (es decir, va a existir dinámica no modelada, por ejemplo, la del actuador, el fenómeno de radiación, etc.) y atendiendo a las prestaciones deseadas para las predicciones del modelo se podrían elegir las cotas para que las predicciones realizadas por los modelos del FPS no generasen errores mayores de 3^oC como máximo y de media no mayores a 1^oC. Por lo tanto $\eta_1 = 3$ y $\eta_2 = 1$ y de esta manera $\hat{\mathbf{\Theta}}_{Pr} = \hat{\mathbf{\Theta}}_P \neq \emptyset$ con lo cual el $FPS_{ide} \neq \emptyset$.

A continuación se procede a determinar el FPS_{ide} mediante el algoritmo $\epsilon - GA$ utilizando los siguientes parámetros[11]:

[9] El resto de parámetros del algoritmo son los mismos que se utilizaron en la optimización de los MOP1...MOP5 del capítulo 3.

[10] El espacio de búsqueda ha sido determinado atendiendo al sentido físico que tienen los parámetros del modelo. El espacio de búsqueda para θ_2 se ha elegido asumiendo que la constante de tiempo del proceso pudiese estar entre 50 y 500 seg. (ver figura 7.9 para observar la dinámica del proceso) y para θ_1 se ha determinado utilizando los rangos de θ_2 y la relación entre ambas variables $\theta_1 = \frac{\theta_2(y(1300) - Ta(1300))}{v(1300)^2}$ teniendo en cuenta que en $t = 1300$ seg. el proceso está en régimen permanente.

[11] El resto de parámetros del algoritmo son los mismos que se utilizaron en la optimización de los OP1...OP5 del capítulo 5.

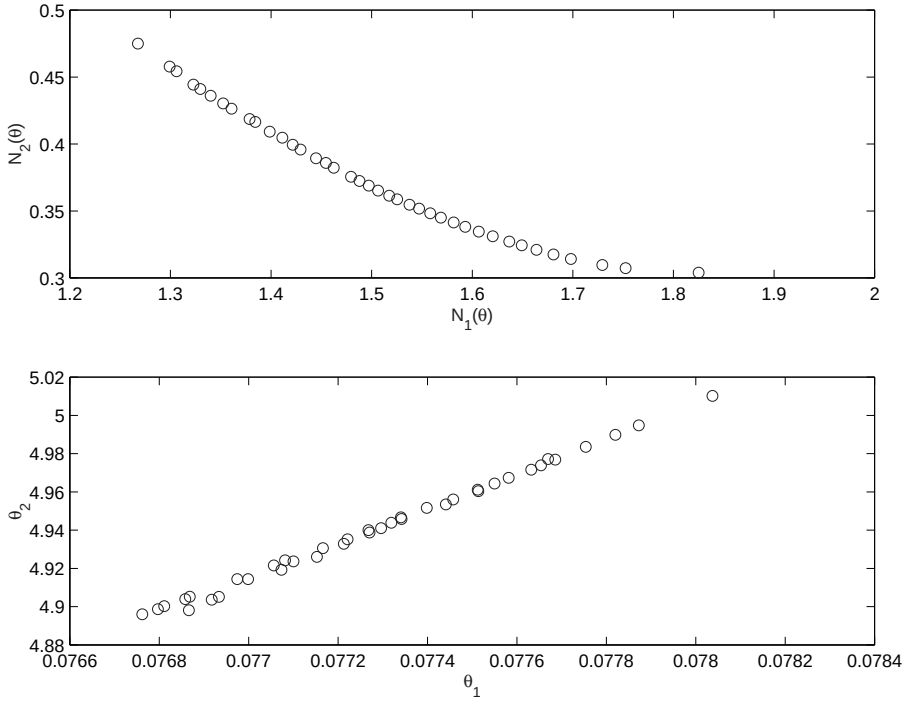

Figura 7.11: Arriba el frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P^*)$. Debajo el conjunto de soluciones óptimas de Pareto $\hat{\mathbf{\Theta}}_P^*$.

- El espacio de búsqueda D es $\theta_1 \in [0.01 \ldots 0.15]$, $k_2 \in [2 \ldots 10.0]$[12].
- $t_{max} = 600$ y $\epsilon = [0.0015, 0.08]$ de manera que el *grid* tenga 100 divisiones en cada dimensión.

La figura 7.12 muestra el resultado del proceso de optimización con ϵ-GA, es decir, el FPS_{ide}^*. Para comprobar el resultado del algoritmo se ha mostrado en la figura el contorno del FPS_1 y el FPS_2, de esta manera también se puede ver como, en este caso, el tener varias normas simultáneas afecta al FPS_{ide} resultante[13].

El algoritmo ha conseguido caracterizar el FPS_{ide} y en especial el ∂FPS_{ide} mediante 63 modelos[14]. El número de evaluaciones de la función $J'(\theta)$ ha sido de 2500, es decir, la cuarta parte de las que hubiesen hecho falta si se hubiese evaluado $J'(\theta)$ en cada *grid*. La media de $J'(\partial FPS_{ide}^*)$ es de 0.030 lo que evidencia la buena convergencia del algoritmo.

La figura 7.13 muestra la respuesta real del proceso $y_{ide}(t)$ junto con la envolvente $\mathcal{Y}(FPS_{ide}^*, U_{ide}(t))$. Se puede observar que la envolvente envuelve $y_{ide}(t)$ de forma muy ajustada. A continuación se procede a validar el FPS_{ide}^* utilizando los datos del experimento que se mostraron en la figura 7.10. Para ello, se simula el modelo del proceso con

[12]El espacio ocupado por los modelos óptimos de proyección obtenidos mediante la optimización multiobjetivo permite reducir el espacio de búsqueda para facilitar la caracterización del FPS_{ide}.

[13]Para calcular FPS_1 y FPS_2 se ha aplicado búsqueda exhaustiva aprovechando que el espacio de búsqueda no es muy grande.

[14]Si se desease obtener una mejor caracterización del FPS_{ide} simplemente habría que reducir ϵ.

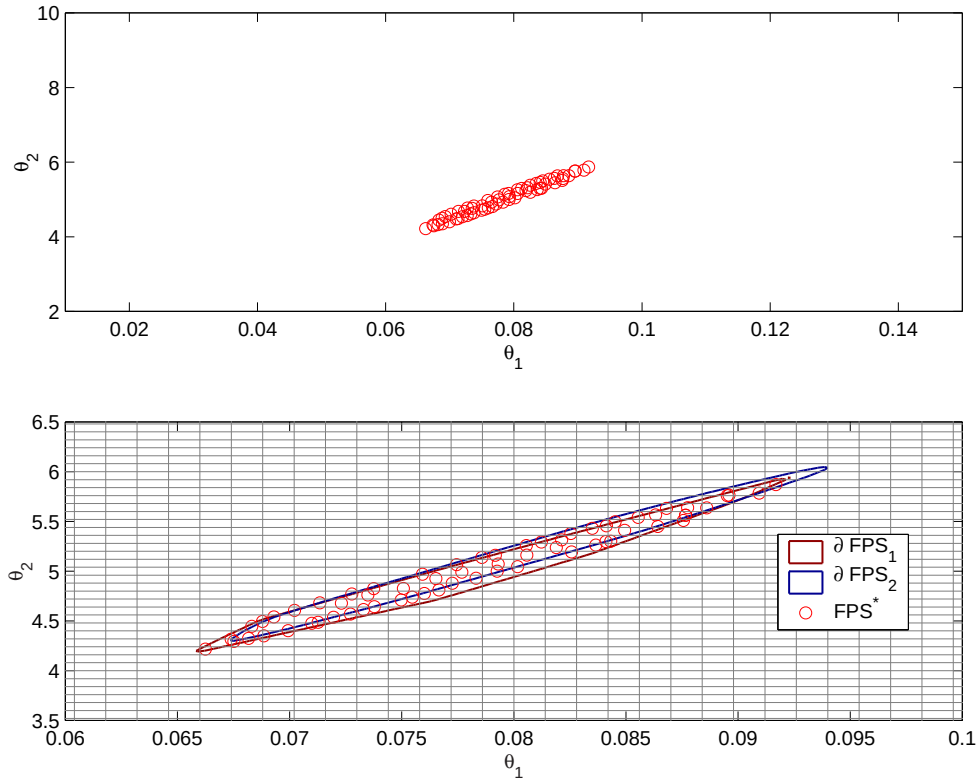

Figura 7.12: Arriba mediante ∘ se representan los modelos $\theta \in FPS^*_{ide}$ dentro del espacio de búsqueda D. Abajo detalle de FPS^*_{ide}, junto con el ∂FPS_1, ∂FPS_2 y el *grid*.

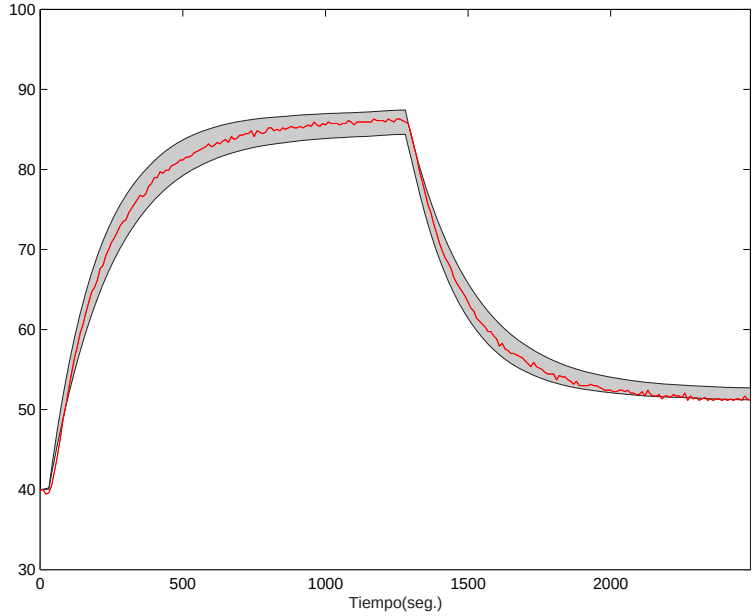

Figura 7.13: $y_{ide}(t)$ y la envolvente de los modelos FPS^*_{ide}.

los parámetros $\theta \in FPS^*_{ide}$. La figura 7.14 muestra la respuesta real del proceso $y_{val}(t)$ y la envolvente $\mathcal{Y}(FPS^*_{ide}, U_{val}(t))$ donde se puede apreciar que la envolvente no envuelve $y_{val}(t)$ en el tramo final del experimento. Por lo tanto, si se utiliza la envolvente como criterio de validación el FPS^*_{ide} quedaría invalidado, sin embargo, si se utiliza como criterio de validación que $FPS^* \neq \emptyset$ el FPS^*_{ide} quedaría validado porque en el FPS^*_{ide} hay modelos consistentes con Ω_{val}, $N_1(\theta)$, $N_2(\theta)$ y sus cotas (ver figura 7.15).

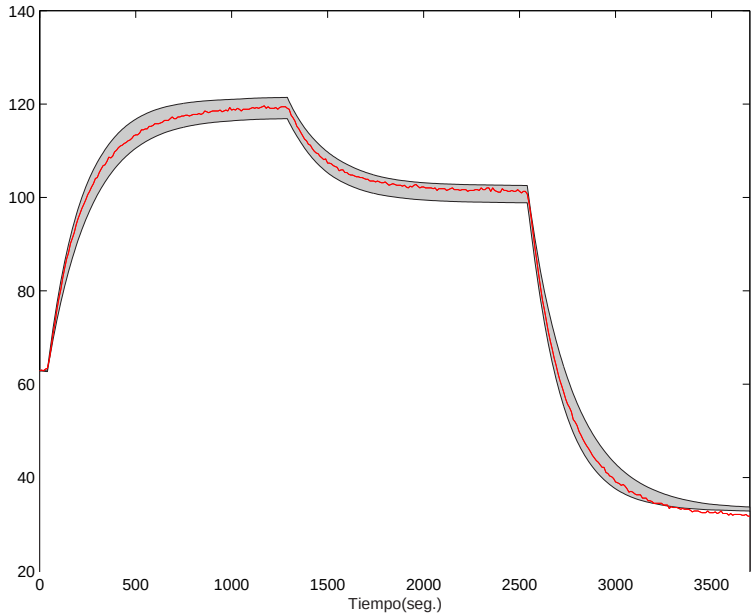

Figura 7.14: $y_{val}(t)$ y la envolvente de los modelos FPS^*_{ide}.

Si se desease validar el FPS^*_{ide} mediante el criterio de la envolvente se podría repetir el proceso aumentando η_1 y η_2 o modificando el modelo. Para que sirva de ejemplo se abordarán estas dos alternativas. La primera se realizará a continuación y la segunda, en la sección siguiente utilizando un modelo de tres parámetros.

Se repite el proceso pero utilizando $\eta_1 = 6$ y $\eta_2 = 2$ (justo el doble que antes), es decir, que las predicciones del modelo no generen errores mayores de 6^oC como máximo y de media no mayores a 2^oC. El FPS^*_{ide} ahora contiene 223 modelos, la figura 7.16 muestra el nuevo FPS^*_{ide} obtenido. La figura 7.17 muestra la respuesta real del proceso $y_{ide}(t)$ y la envolvente $\mathcal{Y}(FPS^*_{ide}, U_{ide}(t))$. Se puede observar que la envolvente envuelve $y_{ide}(t)$ pero ahora de forma más holgada. En cuanto a la validación en la figura 7.18 se muestra la respuesta real del proceso $y_{val}(t)$ y la envolvente $\mathcal{Y}(FPS^*_{ide}, U_{val}(t))$ donde se puede apreciar que la envolvente, aunque de forma muy ajustada, sí que envuelve $y_{val}(t)$. En la figura 7.16 también se muestran, mediante ' $*$ ', los modelos del FPS_{ide} consistentes con Ω_{val} con lo cual $FPS^* \neq \emptyset$ y por lo tanto el FPS^*_{ide} quedaría validado.

Relacionado con los modelos nominales óptimos, una vez obtenido el FPS^*_{ide} se puede determinar el modelo de peor caso $\hat{\theta}^*_c$ como se indica en las ecuaciones (7.18) y (7.16) dando como resultando:

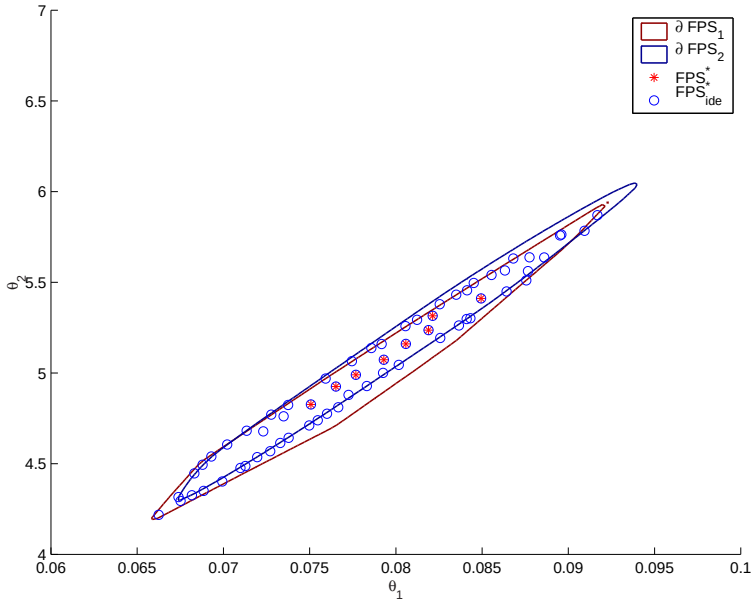

Figura 7.15: FPS^*_{ide}, junto con el ∂FPS_1, ∂FPS_2 y FPS^*. FPS^* contiene los modelos del FPS^*_{ide} consistentes con Ω_{val}. Para $\eta_1 = 3$ y $\eta_2 = 1$.

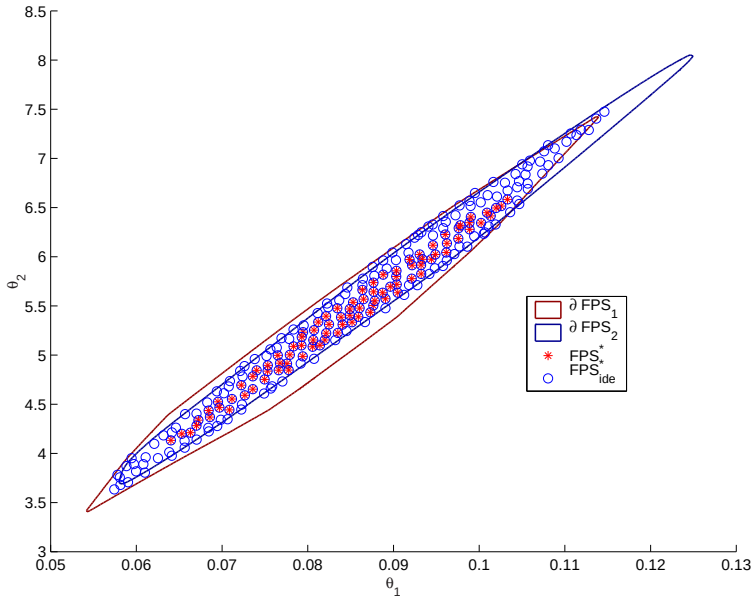

Figura 7.16: FPS^*_{ide}, junto con el ∂FPS_1, ∂FPS_2 y FPS^*. FPS^* contiene los modelos del FPS^*_{ide} consistentes con Ω_{val}. Para $\eta_1 = 6$ y $\eta_2 = 2$.

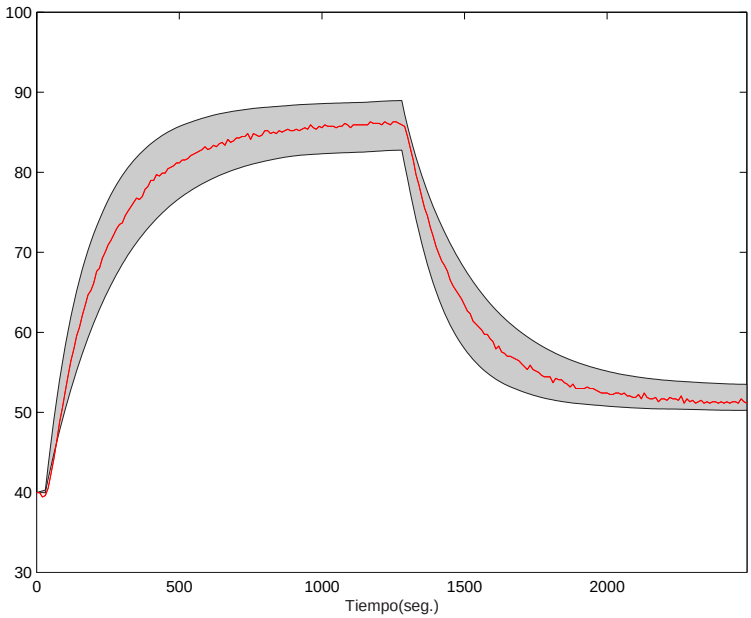

Figura 7.17: $y_{ide}(t)$ y la envolvente de los modelos FPS^*_{ide}. Para $\eta_1 = 6$ y $\eta_2 = 2$.

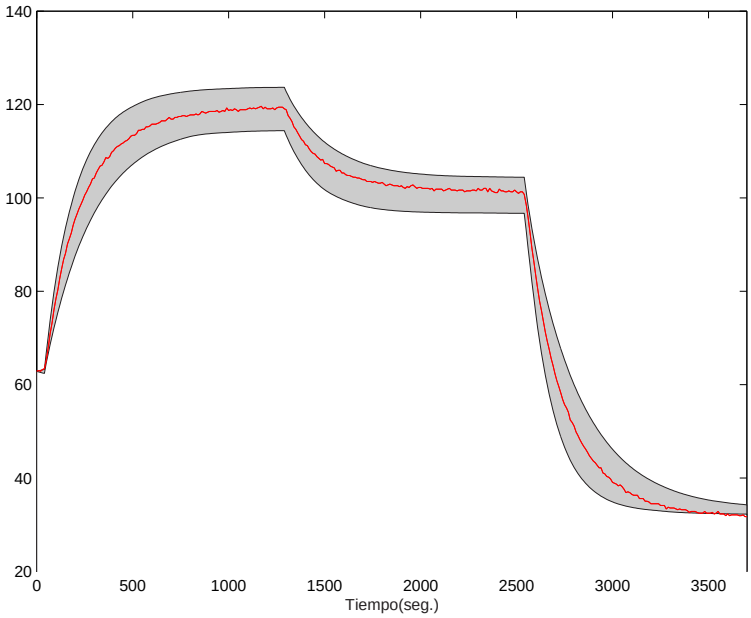

Figura 7.18: $y_{val}(t)$ y la envolvente de los modelos FPS^*_{ide}. Para $\eta_1 = 6$ y $\eta_2 = 2$.

$$\hat{\theta}_c^* = [0.08602, 5.5546]^T.$$

Este modelo produce los siguientes valores para las normas N_1 y N_2:

$$\Omega_{ide} \Rightarrow N_1(\hat{\theta}_c^*) = 2.691^oC,\ N_2(\hat{\theta}_c^*) = 0.680^oC,$$
$$\Omega_{val} \Rightarrow N_1(\hat{\theta}_c^*) = 2.741^oC,\ N_2(\hat{\theta}_c^*) = 1.0519^oC,$$

con lo cual este modelo nominal quedaría también validado, ya que pertenece a $FPS = FPS_{ide} \bigcap FPS_{val}$.

Otro modelo óptimo sería el de proyección interpolada restringida (ver la ecuación (7.20)) que en este caso coincide con el modelo de proyección para la norma N_2

$$\hat{\theta}_{pi}^* = \hat{\theta}_{N_2} = [0.0780, 5.010]^T$$

y que produce los siguientes valores para las normas N_1 y N_2:

$$\Omega_{ide} \Rightarrow N_1(\hat{\theta}_{pi}^*) = 1.802^oC,\ N_2(\hat{\theta}_{pi}^*) = 0.303^oC,$$
$$\Omega_{val} \Rightarrow N_1(\hat{\theta}_{pi}^*) = 1.657^oC,\ N_2(\hat{\theta}_{pi}^*) = 0.604^oC,$$

con lo cual este modelo nominal, además de presentar resultados mejores que $\hat{\theta}_c^*$, quedaría validado, ya que pertenece al FPS.

La figura 7.19 muestra la localización de los modelos $\hat{\theta}_c^*$, $\hat{\theta}_{pi}^*$ junto con ∂FPS_1, ∂FPS_2 y los modelos de proyección restringida $\hat{\Theta}_{Pr}^*$ (obtenidos anteriormente).

Para chequear que $\hat{\theta}_c^*$ es una buena aproximación del centro de Chebyshev del FPS_{ide}, se ha determinado éste[15]

$$\hat{\theta}_c = arg \min_{\theta \in \mathbf{D}} \max_{\overline{\theta} \in \mathbf{FPS_{ide}}} \|\theta - \overline{\theta}\|_2 = [0.086, 5.57].$$

y representado también sobre la figura 7.19.

7.3.2. Modelo con 3 parámetros

Tal y como se comentó en la sección anterior se va a proceder a realizar la IR del sistema térmico adoptando un modelo que, además de modelar las pérdidas de calor por convección y conducción, también incorporará las debidas al fenómeno de radiación [166]:

$$\begin{aligned} \dot{x}(t) &= \frac{1}{1000}\left(\theta_1 v(t)^2 - \theta_2\left(x(t) - Ta(t)\right) - \theta_3\left(\frac{273.0 + x(t)}{100}\right)^4\right), \\ \hat{y}(t) &= x(t), \end{aligned} \tag{7.28}$$

[15]Para ello se ha utilizado el FPS obtenido mediante búsqueda exhaustiva.

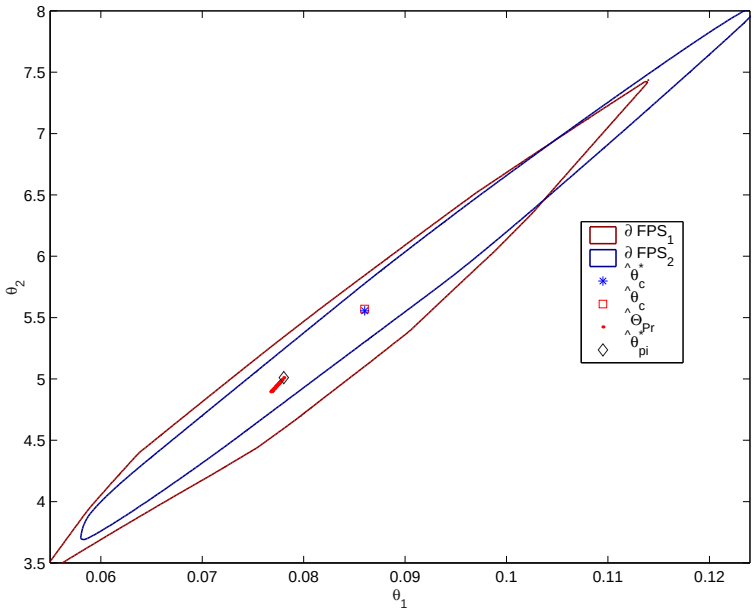

Figura 7.19: Modelos nominales óptimos, ∂FPS_1 y ∂FPS_2. Aproximación al centro de Chebyshev $\hat{\theta}_c^*$, modelo nominal de proyección interpolada restringida $\hat{\theta}_{pi}^*$, modelos de proyección restringida $\hat{\Theta}_{Pr}^*$. Centro de Chebyshev $\hat{\theta}_c^*$.

donde:

- $\dot{x}(t)$ es la variable de estado del modelo,
- $v(t)$ es la tensión de entrada al proceso en voltios $0 \div 100(\,\%) \Rightarrow 0 \div 7.5v$,
- $\hat{y}(t)$ es la temperatura de la salida de proceso en (^oC),
- $Ta(t)$ es la temperatura del aire en (^oC) y
- $\theta = [\theta_1, \theta_2, \theta_3]^T$ son los parámetros del modelo.

Se va a utilizar la misma forma de determinar el estado $x(0)$ y también se utilizarán las mismas normas $N_1(\theta)$ y $N_1(\theta)$ que se utilizaron en el modelo de 2 parámetros. Para escoger las cotas η_1 y η_2, se utilizará la información que genere el nuevo frente de Pareto del siguiente problema de optimización multiobjetivo.

$$\min_{\theta \in D} \mathbf{J}(\theta) = \{N_1, N_2\}. \tag{7.29}$$

Los parámetros del algoritmo $\not\epsilon$MOGA escogidos son los siguientes:

- El espacio de búsqueda D es $\theta_1 \in [0.01 \ldots 0.3]\frac{^oC}{seg.\,\%^2}$, $\theta_2 \in [2.0 \ldots 20.0]\frac{1}{seg.}$, $\theta_3 \in [0.0 \ldots 1.0]\frac{1}{seg.}$.

- $t_{max} = 5000$ y $n\_box_1 = n\_box_2 = 50$.

La figura 7.20 muestra los modelos óptimos de proyección $\hat{\mathbf{\Theta}}_P^*$ obtenidos como solución al problema de optimización multiobjetivo y el frente de Pareto originado. Del análisis del frente de Pareto se puede determinar que los modelos de proyección $\hat{\theta}_{N_1}$ y $\hat{\theta}_{N_2}$ son los siguientes:

$$\hat{\theta}_{N_1} = [0.0766, 4.380, 0.190]^T \Rightarrow N_1(\hat{\theta}_{N_1}) = \mathbf{0.979}^oC,\ N_2(\hat{\theta}_{N_1}) = 0.332^oC,$$
$$\hat{\theta}_{N_2} = [0.0776, 4.528, 0.176]^T \Rightarrow N_1(\hat{\theta}_{N_2}) = 1.461^oC,\ N_2(\hat{\theta}_{N_2}) = \mathbf{0.239}^oC.$$

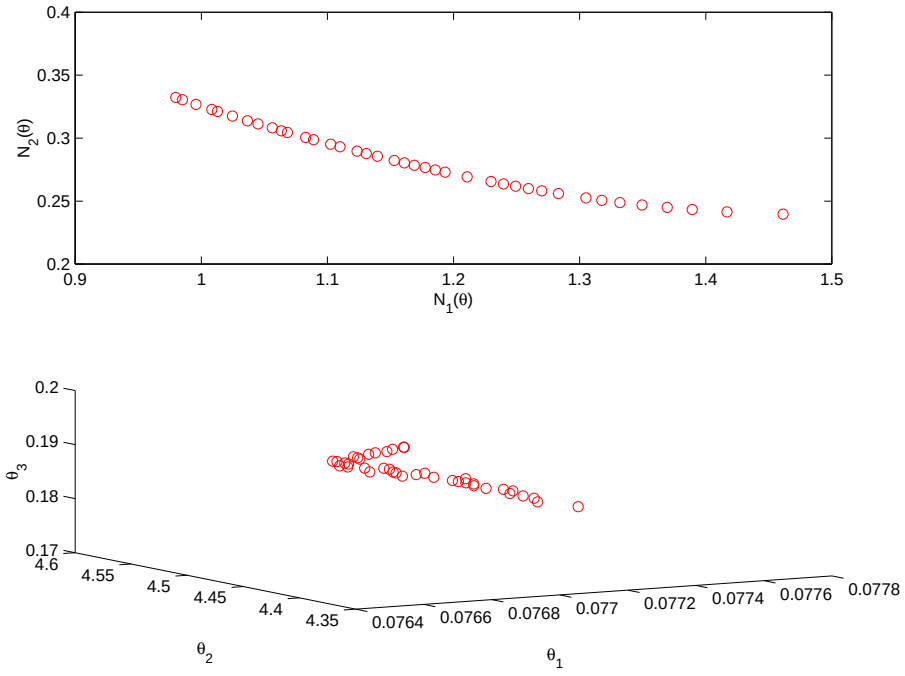

Figura 7.20: Arriba el frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P^*)$. Debajo el conjunto de soluciones óptimas de Pareto $\hat{\mathbf{\Theta}}_P^*$.

Ahora estos modelos presentan mejores prestaciones que los obtenidos en la sección anterior, como es lógico ya que el modelo tiene un parámetro más y puede ajustar mejor la dinámica. Para compara resultados con el modelo de 2 parámetros se van a fijar los mismos valores que antes para $\eta_1 = 3$ y $\eta_2 = 1$. De esta manera, de nuevo, $\hat{\mathbf{\Theta}}_{Pr} = \hat{\mathbf{\Theta}}_P \neq \emptyset$ con lo cual el $FPS_{ide} \neq \emptyset$.

A continuación se procede a determinar el FPS_{ide} mediante el algoritmo $\epsilon - GA$ utilizando los siguientes parámetros:

- El espacio de búsqueda D es $\theta_1 \in [0.01 \ldots 0.15]$, $\theta_2 \in [2 \ldots 10.0]$, $\theta_3 \in [0 \ldots 0.8]$.
- $t_{max} = 4975$ y $\epsilon = [0.0035, 0.2, 0.02]$ de manera que el *grid* tenga 40 divisiones en cada dimensión[16].

[16]Se ha reducido el número de divisiones por dimensión para evitar que el FPS_{ide} contenga un número excesivo de modelos.

La figura 7.21 muestra el resultado del proceso de optimización con ϵ-GA, es decir, el FPS^*_{ide}. El algoritmo ha caracterizado el FPS_{ide} mediante 376 modelos. El número de evaluaciones de la función $J'(\theta)$ ha sido de 20000, es decir, la cuarta parte de las que hubiesen hecho falta si se hubiese evaluado $J'(\theta)$ en cada *box* del *grid*[17]. La media de $J'(\partial FPS^*_{ide})$ es de 0.044 lo que demuestra, de nuevo, la buena convergencia del algoritmo.

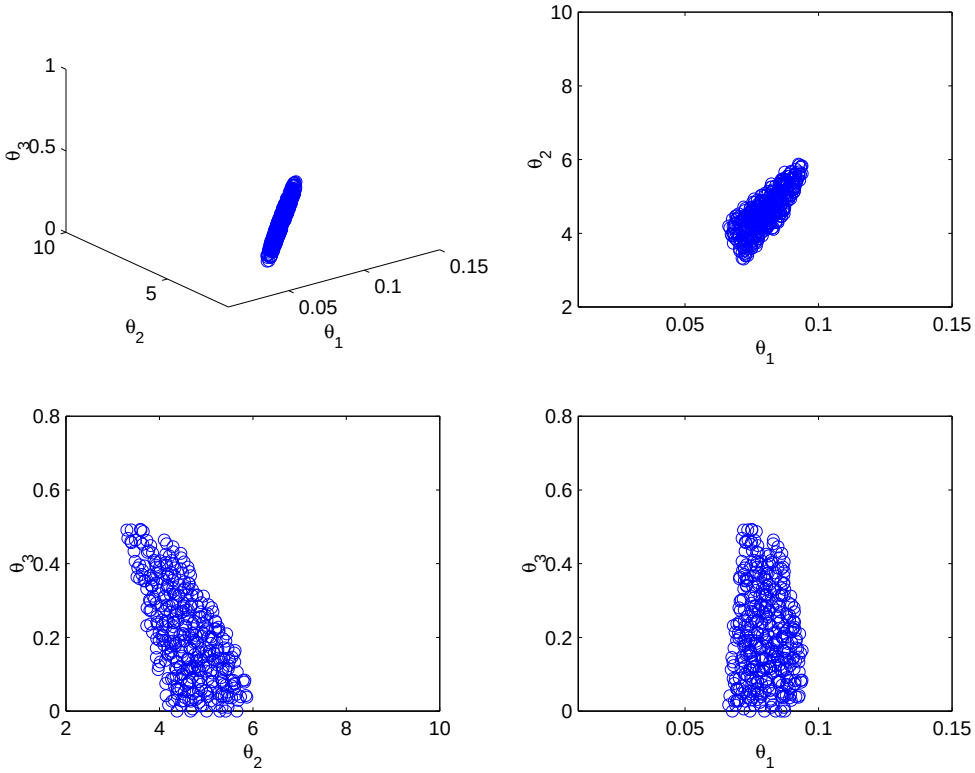

Figura 7.21: Mediante ∘ se representan los modelos $\theta \in FPS^*_{ide}$ dentro del espacio de búsqueda D y sus respectivas proyecciones sobre los planos $\{\theta_1, \theta_2\}$, $\{\theta_2, \theta_3\}$ y $\{\theta_1, \theta_3\}$.

Las figuras 7.22 y 7.23 muestran la respuesta real del proceso $y_{ide}(t)$ y la envolvente $\mathcal{Y}(FPS^*_{ide}, U_{ide}(t))$ y la respuesta real del proceso $y_{val}(t)$ y la envolvente $\mathcal{Y}(FPS^*_{ide}, U_{val}(t))$ respectivamente. Se puede observar que, en ambos casos, las envolventes envuelven las respuestas $y_{ide}(t)$ e $y_{val}(t)$ sin problemas[18]. Por lo tanto, si se utiliza la envolvente como criterio de validación el FPS^*_{ide} quedaría validado. Si se utiliza como criterio de validación que $FPS^* \neq \emptyset$ el FPS^*_{ide} también quedaría validado porque el FPS^*_{ide} contiene 258 modelos consistentes con Ω_{val}, es decir el 71 % de sus modelos.

Una vez obtenido y validado el FPS^*_{ide} se procede a determinar el modelo de peor caso $\hat{\theta}^*_c$ [19] y el modelo de proyección interpolada restringida[20].

[17] Además con la búsqueda exhaustiva la caracterización del FPS_{ide} hubiese sido peor.
[18] Esto permitiría plantearse la posibilidad de reducir de las cotas η_1 y η_2 para reducir el FPS_{ide}.
[19] Para su obtención se ha utilizado un GA clásico.
[20] Vuelve a coincidir con el modelo de proyección para la norma N_2.

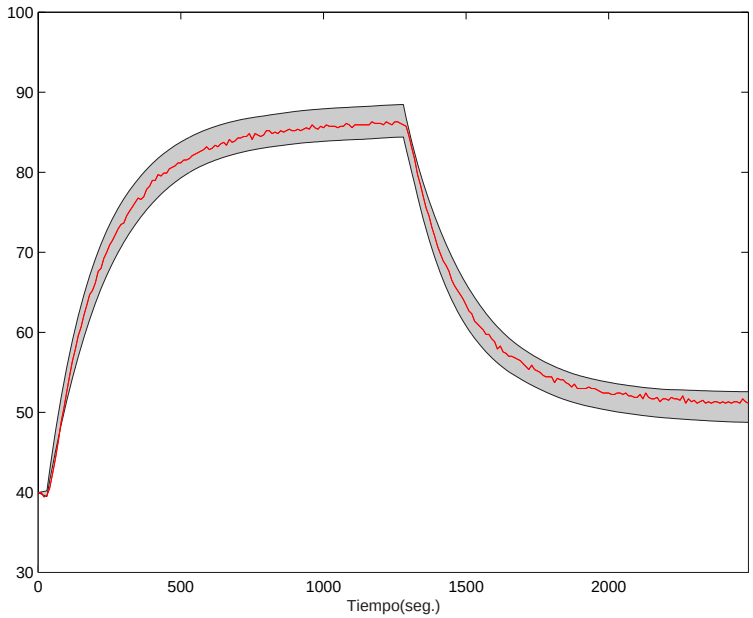

Figura 7.22: $y_{ide}(t)$ y la envolvente de los modelos FPS^*_{ide}.

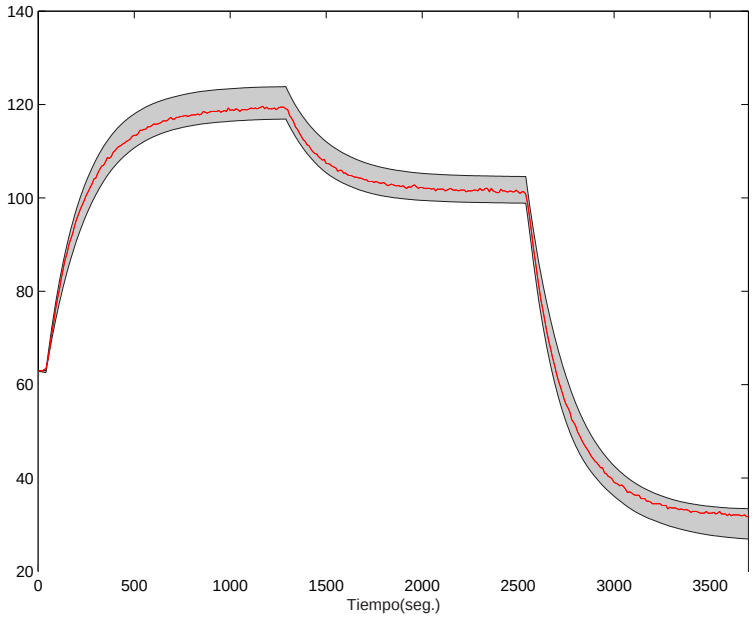

Figura 7.23: $y_{val}(t)$ y la envolvente de los modelos FPS^*_{ide}.

$$\hat{\theta}_c^* = [0.079670, 4.486096, 0.245915]^T,$$

$$\hat{\theta}_{pi}^* = \hat{\theta}_{N_2} = [0.0776, 4.528, 0.176]^T.$$

Estos modelos producen los siguientes valores para las normas N_1 y N_2:

$$\Omega_{ide} \Rightarrow N_1(\hat{\theta}_c^*) = 1.551^oC,\ N_2(\hat{\theta}_c^*) = 0.411^oC,$$

$$\Omega_{val} \Rightarrow N_1(\hat{\theta}_c^*) = 1.703^oC,\ N_2(\hat{\theta}_c^*) = 0.721^oC,$$

$$\Omega_{ide} \Rightarrow N_1(\hat{\theta}_{pi}^*) = 1.461^oC,\ N_2(\hat{\theta}_{pi}^*) = 0.239^oC,$$

$$\Omega_{val} \Rightarrow N_1(\hat{\theta}_{pi}^*) = 1.655^oC,\ N_2(\hat{\theta}_{pi}^*) = 0.5279^oC,$$

con lo cual ambos modelos quedan validados, ya que pertenecen a $FPS = FPS_{ide} \bigcap FPS_{val}$.

La figura 7.24 muestra la localización de los modelos $\hat{\theta}_c^*$, $\hat{\theta}_{pi}^*$ junto con el FPS_{ide}^* y los modelos de proyección restringida $\hat{\Theta}_{Pr}^*$ (obtenidos anteriormente).

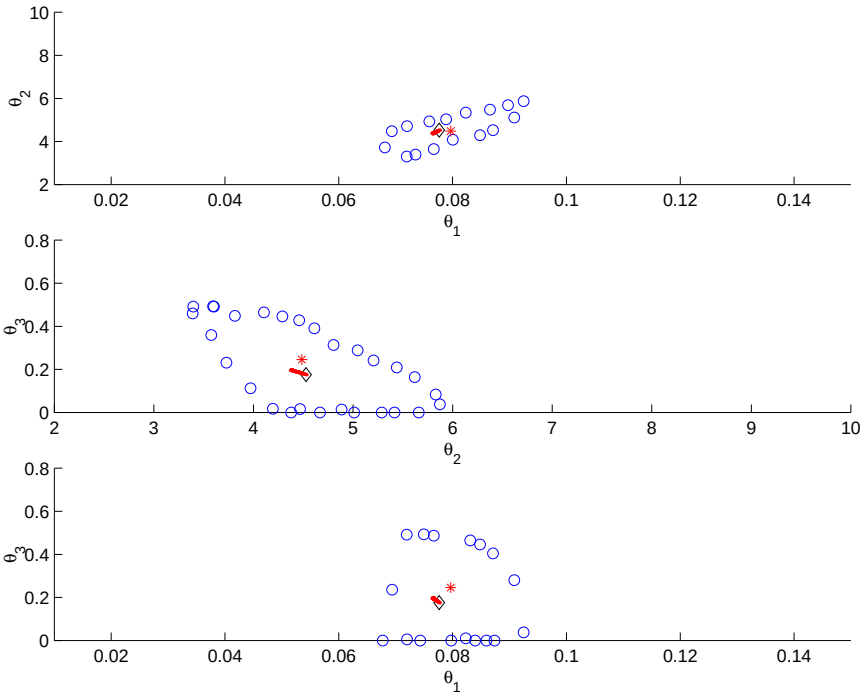

Figura 7.24: Modelos nominales óptimos. (*) Aproximación al centro de Chebyshev $\hat{\theta}_c^*$, ($\diamond$) modelo nominal de proyección interpolada restringida $\hat{\theta}_{pi}^*$ y ($\cdot$) modelos de proyección restringida $\hat{\Theta}_{Pr}^*$.

7.4. Ejemplo 2. Modelo biomédico

El objetivo de este apartado es abordar la IR de los parámetros un modelo que refleja la interacción periódica de un determinado fármaco con un cierto canal iónico en células cardíacas. El fármaco es capaz de bloquear el canal, modificando así su potencial de acción (PA) ayudando, de esta manera, a corregir ciertas patologías que alteran el funcionamiento normal cardíaco[21].

El PA es la diferencia de potencial entre el medio intracelular y el extracelular, es decir, entre ambos lados de la membrana iónica. La membrana permite el intercambio de iones entre ambos medios a través de los canales iónicos y las bombas electrogénicas generando las corrientes iónicas de sodio (Na^+), calcio (Ca^{2+}), potasio (K^+) y otras sustancias responsables de generar el PA.

Hay ciertas células que pueden ser excitadas mediante estímulos despolarizantes (p.e. las cardíacas) modificando así la conductancia de la membrana y creando un transitorio en su potencial como por ejemplo el que se muestra en la figura 7.25 para una célula miocárdica ventricular.

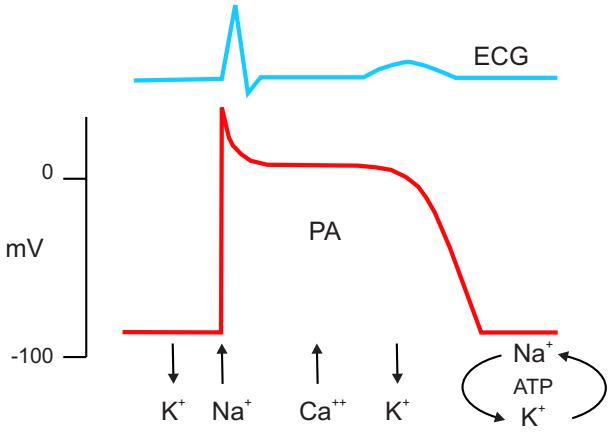

Figura 7.25: Representación del PA de una célula miocárdica ventricular y de las corrientes iónicas asociadas más importantes. Las flechas hacia abajo son corrientes salientes y las que van hacia arriba entrantes.

El PA podría describirse de forma muy simplifica de la siguiente manera. Partiendo de una fase de reposo donde el potencial tiene -80mV, ante la presencia de un estímulo despolarizante que haga que el PA llegue a un determinado umbral, se abren los canales de sodio y se produce la despolarización llegando a la meseta del PA (20mV durante aproximadamente 200mseg.) donde el PA se equilibra con la entrada y salida de iones calcio y potasio respectivamente. A continuación, al inactivarse los canales de calcio se produce la repolarización de forma progresiva hasta llegar de nuevo al reposo o periodo refractario en el que la célula permanece inexcitable y cuya duración depende de la intensidad del

[21] Este trabajo se circumscribe, dentro de un proyecto de colaboración con el Departamento de Ingeniería Electrónica de la UPV para la estimación de parámetros en modelos de bloqueo de fármacos.

estímulo. Durante esta última fase se activan las bombas electrogénicas y los intercambiadores iónicos para restablecer las concentraciones que se han visto alteradas durante el desarrollo del PA.

Algunas células del corazón tienen actividad automática con una determinada frecuencia. Esta actividad genera impulsos eléctricos que se propagan a través del tejido y estimulan el PA en el resto de células provocando así la contracción de cada una de las cámaras del corazón.

Existen muchas alteraciones patológicas del funcionamiento del corazón y, por suerte, fármacos que ayudan prevenirlas. Estos fármacos actúan de diferentes formas, por ejemplo, algunos aumentan la contracción del miocardio (cardiotónicos), otros modifican el flujo de sangre (vasodilatadores) y otros se encargan de alterar el ritmo cardíaco (antiarrítmicos).

Estos últimos se caracterizan por actuar sobre el PA modificándolo en alguna de sus cuatro fases, para ello, lo que hacen es bloquear, en parte, alguna de las corrientes iónicas (por ejemplo la de sodio). Así, se puede prolongar o acortar el PA, reducir las despolarizaciones, modificar el umbral de disparo del PA, modificar el potencial de reposo, aumentar el periodo refractario, etc. actuando de esta manera sobre patologías como arritmias, taquicardias, elevando el umbral de fibrilación, produciendo un efecto inotrópico, etc.

A la hora de modelar la interacción del fármaco con el receptor (canal iónico) existen muchas posibilidades (ver [27] y sus referencias) aunque la que se va a adoptar en esta tesis es la hipótesis *Guarded Receptor* (GRT). La configuración del canal depende del PA pudiendo encontrarse accesible para poder ser bloqueado por el fármaco o inaccesible (debido a la existencia de compuertas en el canal que impiden su bloqueo). La figura 7.26 ilustra las tres situaciones posibles de configuración del canal, inaccesible I, no podría ser bloqueado por el fármaco, accesible A permitiendo que el fármaco bloquease el canal B.

$$I \underset{\beta}{\overset{\alpha}{\leftrightharpoons}} A \underset{l}{\overset{kD}{\leftrightharpoons}} B. \tag{7.30}$$

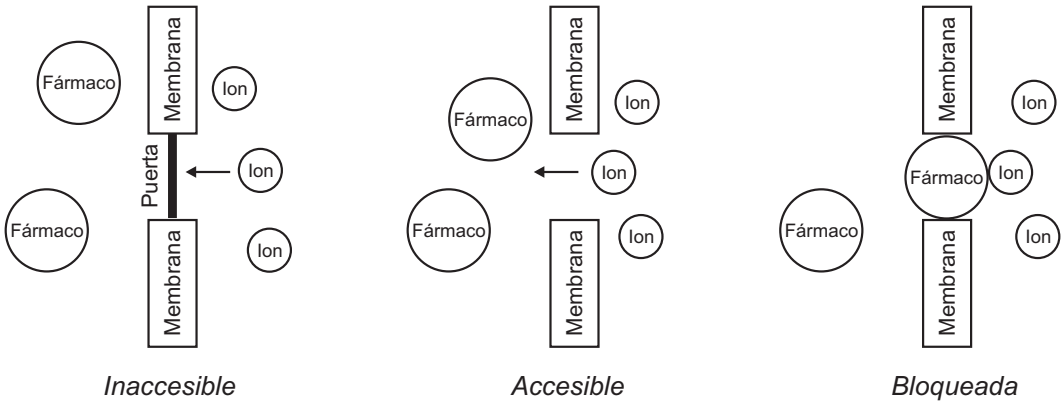

Figura 7.26: Modelo GRT de interacción del fármaco con los canales y las puertas que impiden el acceso a éstos.

Los parámetros α y β dependen de la excitación de la célula, si se encuentra en reposo el canal tiende al estado de inaccesibilidad, mientras que si está activo favorece la accesibilidad permitiendo la interacción entre el canal y el fármaco D produciendo el bloqueo del primero con un ratio k (constante real de asociación). Los canales bloqueados pueden también disociarse en función de l (constante real de disociación). k y l son considerados independientes de la excitación de la célula.

La dinámica de la interacción entre I y A es rápida comparada con la de la interacción entre A y B. Para un pulso de estimulación, el comportamiento del canal puede ser descrito por un proceso de dos pasos, involucrando un estado de activación (intervalo de activación t_a) y un estado de recuperación del canal (intervalo de recuperación t_r)

$$U + D \underset{g_i l}{\overset{f_i k}{\rightleftharpoons}} B, \tag{7.31}$$

donde i representa el estado (activación a o recuperación r) y U los canales no bloqueados (accesibles o no). f_i y g_i son la fracción de sitios accesibles y fracción de canales bloqueados por el fármaco disponibles para disociarse respectivamente y son considerados sensibles al estímulo. Por lo tanto, k y l serían la relación real de asociación y disociación mientras que las relaciones aparentes serían $K_i = f_i k$ y $L_i = g_i l$. Estas relaciones aparentes son sensibles a la naturaleza del protocolo de estimulación [150]. Por esta razón es necesario tener en cuenta el protocolo de estimulación para la obtención de las constantes aparentes, ya que el objetivo es poder comparar fármacos a través de éstas.

La evolución temporal de la fracción de canales bloqueados B por el fármaco se ajusta a la dinámica de un sistema de primer orden de la siguiente manera:

$$\frac{db}{dt} = K_i D(1 - b) - L_i b. \tag{7.32}$$

La figura 7.27 muestra la evolución de $b(t)$ en función del intervalo (activación, recuperación) donde se puede observar que sigue una evolución exponencial, típica del sistema de primer orden, para cada intervalo o fase del estímulo. La ecuación (7.32) tiene como solución:

$$b(t) = b_{i,\infty} + (b_0 - b_{i,\infty})e^{-\lambda_i t}, \tag{7.33}$$

donde $b_{i,\infty} = \frac{K_i D}{K_i D + L_i}$ sería la fracción de canales bloqueados en $t = \infty$, $\lambda_i = K_i D + L_i$ correspondería a la constante de tiempo y b_0 sería la fracción de de canales bloqueados en el instante inicial, $t = 0$.

La fracción de canales bloqueados justo antes de que empiece cada intervalo puede ser descrita por una secuencia de ecuaciones recurrentes de la siguiente manera:

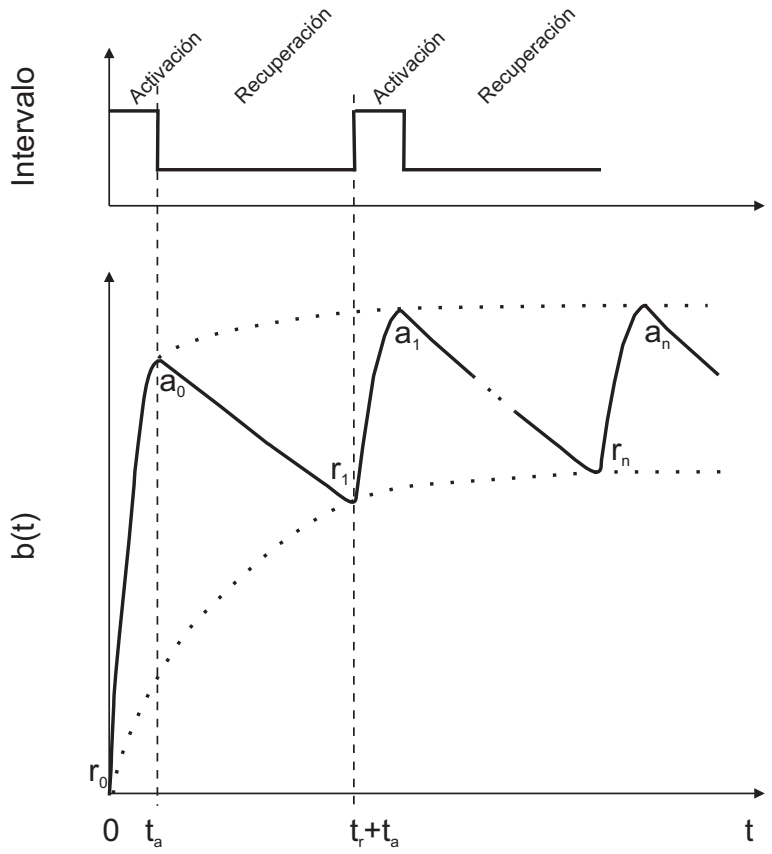

Figura 7.27: Evolución de $b(t)$ función del intervalo (activación, recuperación). Sobre $b(t)$ se observan los puntos a_n y r_n que representan la fracción de canales bloqueados justo antes de que empiece cada intervalo. La unión de los puntos a_n y r_n también es una exponencial.

$$r_n = a_{n-1}e^{-\lambda_r t_r} + r_\infty(1 - e^{-\lambda_r t_r}), \tag{7.34}$$
$$a_n = r_n e^{-\lambda_a t_a} + a_\infty(1 - e^{-\lambda_a t_a}), \tag{7.35}$$

donde r_n y a_n representan $b(t)$ en $t = nt_a$ y $t = n(t_a + t_r)$ respectivamente y

$$\lambda_r = K_r D + L_r, \tag{7.36}$$
$$r_\infty = \frac{K_r D}{K_r D + L_r}, \tag{7.37}$$
$$\lambda_a = K_a D + L_a, \tag{7.38}$$
$$a_\infty = \frac{K_a D}{K_a D + L_a}, \tag{7.39}$$

siendo K_r, L_r, K_a y L_a las relaciones aparentes de asociación/disociación para cada uno de los dos intervalos (activación, recuperación). r_n y a_n siguen una trayectoria exponencial. Analizando simplemente r_n su trayectoria puede ser descrita de la siguiente manera (para

ello se han combinado las ecuaciones 7.34 y 7.35 entre sí):

$$r_n = r_{ss} + (r_0 - r_{ss})e^{-n\lambda}, \tag{7.40}$$

donde:

$$r_{ss} = a_\infty + \gamma_r(r_\infty - a_\infty), \tag{7.41}$$
$$\lambda = \lambda_a t_a + \lambda_r t_r, \tag{7.42}$$
$$\gamma_r = \frac{1 - e^{-\lambda_r t_r}}{1 - e^{-\lambda}}. \tag{7.43}$$

7.4.1. Identificación Robusta

Una vez modelado el comportamiento de bloqueo de canales iónicos mediante fármacos (ecuaciones (7.36) a (7.43)) se procede a abordar la IR para el caso particular de interacción del fármaco antiarrítmico cibenzolina con el canal sodio en células del músculo cardíaco utilizando la información del ejemplo presentado en [150].

Es posible estimar la fracción de canales bloqueados observando como el bloqueo modifica la conductancia del canal y por lo tanto la corriente de sodio I. Dado un potencial de la membrana V

$$I = g(1 - b)V.$$

Debido a que es difícil medir I, se puede utilizar la derivada máxima del potencial de membrana $\dot{V}_{max}$ pues se considera proporcional a I[22]. Se toman medidas de $\dot{V}_{max}$ justo al inicio del intervalo de activación. Por lo tanto, las medidas seguirán una secuencia de valores exponencial proporcional a la reflejada en (7.40):

$$\dot{V}_{max,n} = \dot{V}_{ss} + (\dot{V}_0 - \dot{V}_{ss})e^{-n\lambda}. \tag{7.44}$$

La relación de proporcionalidad entre $\dot{V}_{max}$ y $b_n = r_n$ se puede determinar mediante:

$$b_n = 1 - \frac{\dot{V}_{max,n}}{\dot{V}_c}, \tag{7.45}$$

siendo $\dot{V}_c$ la observación de $\dot{V}_{max}$ realizada en ausencia del fármaco.

La tabla 7.1 muestra los datos que se utilizan en el proceso de identificación. Se utiliza una concentración de $D = 16\mu M$ de cibenzolina un $t_a = 1mseg.$ y $t_r = 0.5, 1.0, 1.5, 2.0$ y $2.5seg.$ en diferentes experimentos. En la figura 7.28 se muestra su representación gráfica. Los datos han sido separados en dos grupos Ω_{ide} que contiene los datos de las columnas 1 y 5 de la tabla 7.1 y que serán utilizados en el proceso de identificación del FPS y Ω_{val} columnas 2,3 y 4 que se utilizarán en la validación del FPS.

[22]Hay que tener en cuenta que la membrana actúa como un condensador $C_m \frac{dV}{dt} = I$ y que la corriente máxima de sodio (y por lo tanto $\dot{V}_{max}$) se produce en el instante inicial de activación del PA.

n	$\dot{V}_{max}$				
0	185	188	192	194	190
1	158	170	173	173	174
2	151	160	164	162	168
3	143	147	158	164	157
4	136	148	153	161	162
5	125	138	144	155	153
6	122	131	144	155	157
7	120	132	143	156	153
8	117	129	143	153	154
9	115	129	139	149	154
10	112	130	137	152	156
11	111	124	138	151	153
12	108	123	139	150	153
13	106	121	134	149	153
14	106	124	134	150	150
15	105	121	135	149	153
16	105	124	137	150	153
17	104	126	132	149	157
18	104	123	136	150	158
19	103	119	137	149	154
$t_r(seg.)$	0.5	1.0	1.5	2.0	2.5
$\dot{V}_c$	199	201	203	203	203

Tabla 7.1: Tabla con los datos de $\dot{V}_{max}$ y $\dot{V}_c$ para diferentes valores de t_r obtenidos de [150].

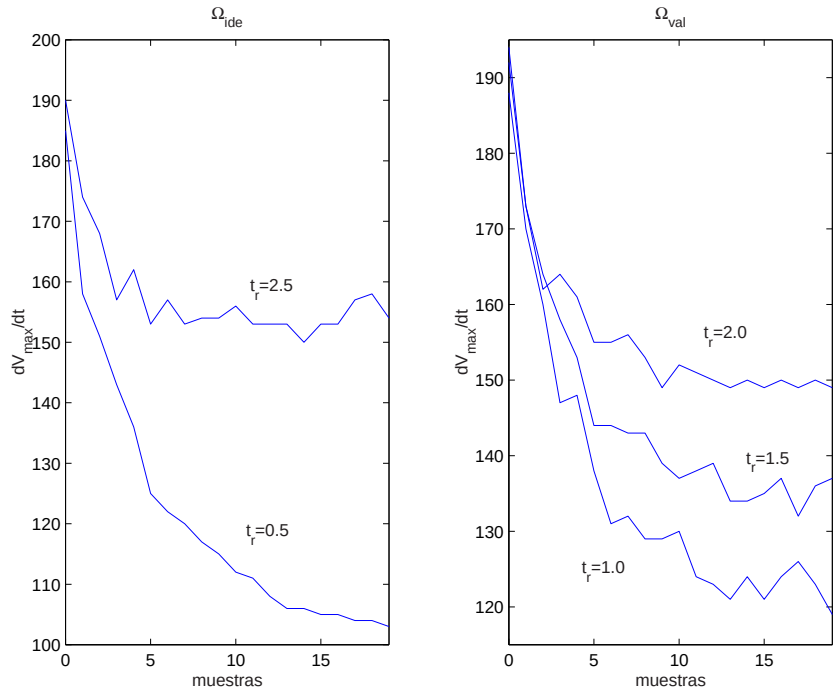

Figura 7.28: Representación de los datos de la tabla 7.1. Se han separado en dos bloques Ω_{ide} columnas $t_r = 0.5$ y $t_r = 2.5$ y Ω_{val} columnas $t_r = 1.0$, $t_r = 1.5$ y $t_r = 2.0$.

En este caso el modelo no viene expresado mediante una ecuación diferencial, ya que fue resuelta de forma explícita, por lo tanto no aparecen variables de estado. La figura 7.29 muestra la estructura E/S del modelo donde:

- $\dot{V}_{max}$ es la derivada máxima del potencial de membrana en V/seg.
- t_a y t_r son los intervalos de activación y recuperación respectivamente.
- D es la concentración del fármaco en M (moles/litro).
- $\theta = [K_a, L_a, K_r, L_r]^T$ son los parámetros del modelo correspondiendo a las relaciones aparentes de asociación/disociación para cada uno de los dos intervalos (activación, recuperación). El modelo es no lineal respecto de θ.

Figura 7.29: Estructura E/S del modelo de bloqueo de fármacos.

Para poder determinar $\dot{V}_{max}$, es necesario establecer $\dot{V}_o$. En este caso se va a tomar $\dot{V}_o = \dot{V}_{max}(0)$. Para determinar el FPS_{ide} se van a utilizar la norma infinito $N_1(\theta)$ y la norma absoluta $N_2(\theta)$ simultáneamente sobre los datos Ω_{ide}.

$$N_1(\theta) = \|\mathbf{e}(\theta, \Omega_{ide})\|_\infty^{\mathbf{w}}, \quad \mathbf{w}_i = 1 \; \forall i \in [1, \ldots, N_{\Omega_{ide}}], \tag{7.46}$$

$$N_2(\theta) = \|\mathbf{e}(\theta, \Omega_{ide})\|_1^{\mathbf{w}}, \quad \mathbf{w}_i = \frac{1}{N_{\Omega_{ide}}} \; \forall i \in [1, \ldots, N_{\Omega_{ide}}], \tag{7.47}$$

Para escoger las cotas η_1 y η_2 de dichas normas y asegurar que con las cotas escogidas será posible validar el FPS_{ide} con los datos Ω_{val} se utilizará la información que genere el frente de Pareto del siguiente problema de optimización multiobjetivo.

$$\min_{\theta \in D} \mathbf{J}(\theta) = \{N_3, N_2, N_4\}, \tag{7.48}$$

donde[23]:

$$N_3(\theta) = \|\mathbf{e}(\theta, \{\Omega_{ide}, \Omega_{val}\})\|_\infty^{\mathbf{w}}, \quad \mathbf{w}_i = 1 \; \forall i \in [1, \ldots, N_{\{\Omega_{ide}, \Omega_{val}\}}], \tag{7.49}$$

[23]Hay que tener en cuenta que $\|\mathbf{e}(\theta, \{\Omega_{ide}, \Omega_{val}\})\|_\infty^{\mathbf{w}} = \text{máx}(\|\mathbf{e}(\theta, \Omega_{ide})\|_\infty^{\mathbf{w}}, \|\mathbf{e}(\theta, \Omega_{val})\|_\infty^{\mathbf{w}})$, por lo tanto $\|\mathbf{e}(\theta, \{\Omega_{ide}, \Omega_{val}\})\|_\infty^{\mathbf{w}} \geq \|\mathbf{e}(\theta, \Omega_{ide})\|_\infty^{\mathbf{w}}$ y $\|\mathbf{e}(\theta, \{\Omega_{ide}, \Omega_{val}\})\|_\infty^{\mathbf{w}} \geq \|\mathbf{e}(\theta, \Omega_{val})\|_\infty^{\mathbf{w}}$ de ahí que sea suficiente con utilizar la norma $N_3(\theta)$ con los datos Ω_{ide} y Ω_{val} de forma conjunta para determinar una cota máxima sobre la norma-∞ con los datos Ω_{ide} y Ω_{val} tratados por separado.

$$N_4(\theta) = \|\mathbf{e}(\theta, \Omega_{val})\|_1^{\mathbf{w}}, \quad \mathbf{w}_i = \frac{1}{N_{\Omega_{val}}} \; \forall i \in [1, \ldots, N_{\Omega_{val}}]. \tag{7.50}$$

Los parámetros del algoritmo $\not\epsilon$MOGA escogidos son los siguientes:

- El espacio de búsqueda es $K_a \in [0.0001 \ldots 1e8]\frac{M}{seg.}$, $L_a \in [0.0001 \ldots 1000]\frac{1}{seg.}$, $K_r \in [0.0001 \ldots 10000]\frac{M}{seg.}$ y $L_r \in [0.0001 \ldots 0.5]\frac{1}{seg.}$.
- $t_{max} = 40000$ y $n\_box_1 = n\_box_2 = n\_box_3 = 100$.

La figura 7.30 muestra el frente de Pareto correspondiente a los modelos óptimos de proyección $\hat{\mathbf{\Theta}}_P^*$ obtenidos como solución al problema de optimización multiobjetivo planteado.

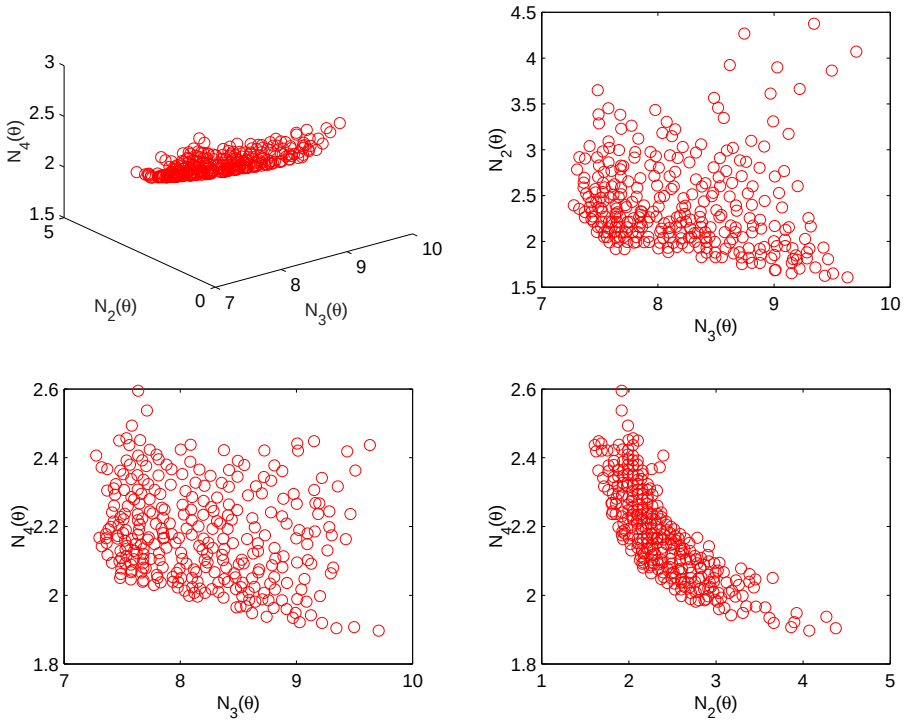

Figura 7.30: Arriba a la izquierda el frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P^*)$. El resto de figuras son su proyección sobre los planos $(N_3(\theta), N_2(\theta))$, $(N_3(\theta), N_4(\theta))$ y $(N_2(\theta), N_4(\theta))$.

A partir del análisis del frente de Pareto se escogen las cotas para que las predicciones realizadas por los modelos del FPS_{ide} no generen errores mayores de $8V/seg.$ como máximo y de media no mayores a $2.5V/seg.$, es decir, $\eta_1 = 8$ y $\eta_2 = 2.5$. De esta manera, se consigue un $\hat{\Theta}_{Pr} \neq \emptyset$ (ver la proyección $(N_3(\theta), N_2(\theta))$ del frente de Pareto) y por lo tanto un $FPS \neq \emptyset$. Además, se asegura que el FPS_{ide} resultará validado pues existirán modelos en el FPS_{ide} consistentes con Ω_{val} y las cotas η_1 y η_2 (ver la proyección $(N_3(\theta), N_4(\theta))$ del frente de Pareto).

A continuación, se procede a determinar el FPS_{ide} mediante el algoritmo $\epsilon - GA$ utilizando los siguientes parámetros:

- El espacio de búsqueda es el mismo que el utilizado con el algoritmo $\not\epsilon$-MOGA.
- $t_{max} = 40000$ y $\epsilon = [1e6, 10, 100, 0.005]$ de manera que el *grid* tenga 100 divisiones en cada dimensión.

La figura 7.31 muestra el resultado del proceso de optimización con ϵ-GA, es decir, el FPS^*_{ide}. El algoritmo ha caracterizado el FPS_{ide} mediante 927 modelos y la media de $J'(\partial FPS^*_{ide})$ es de 0.00942 lo que demuestra una muy buena convergencia del algoritmo (la media de $J'(\partial FPS^*_{ide})$ ideal sería 0).

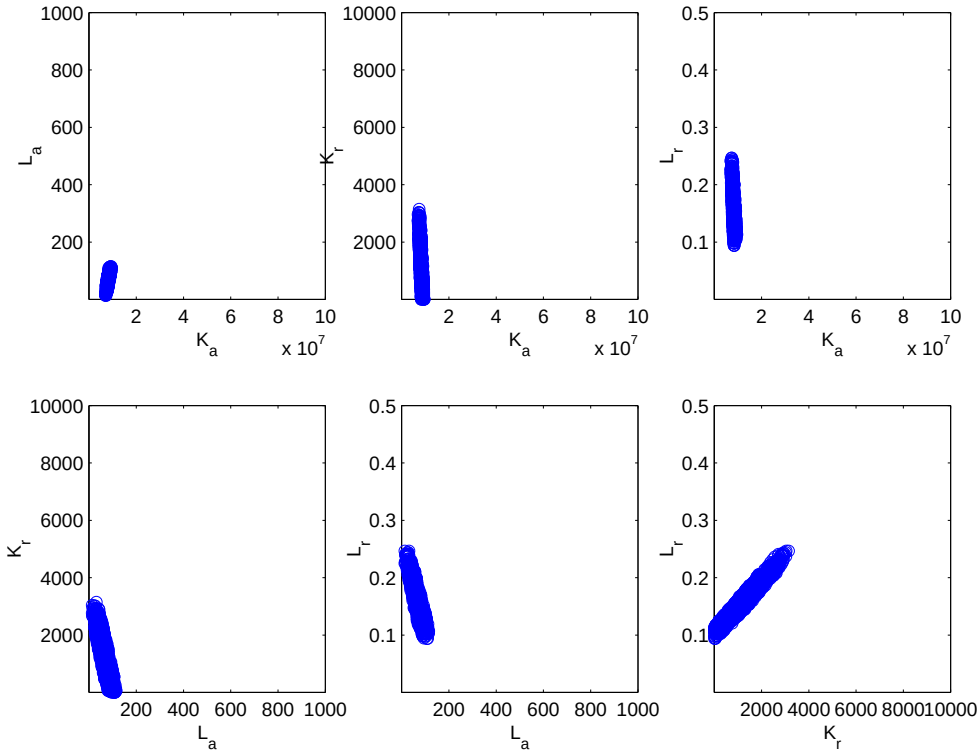

Figura 7.31: Mediante ∘ se representan las proyecciones de los modelos del FPS^*_{ide} dentro del espacio de búsqueda sobre los planos $\{K_a, L_a\}$, $\{K_a, K_r\}$, $\{K_a, L_r\}$, $\{L_a, K_r\}$, $\{L_a, L_r\}$ y $\{K_r, L_r\}$.

La figura 7.32 muestra los datos Ω_{ide}, junto con la envolvente producida por el FPS^*_{ide}, mientras que la figura 7.33 muestran los datos Ω_{val}, junto con la envolvente producida por el FPS^*_{ide}. El criterio que se va a utilizar para la validación es el que determina si en el FPS_{ide} hay datos consistentes con Ω_{val} y las cotas establecidas ($FPS \neq \emptyset$). El FPS^*_{ide} quedaría validado porque el FPS^*_{ide} contiene 19 modelos consistentes con Ω_{val}.

A continuación, una vez validado el FPS^*_{ide} se procede a determinar el modelo nominal de proyección interpolada restringida $\hat{\theta}^*_{pi}$, para ello se determina primero el modelo de peor caso $\hat{\theta}^*_c$.

$$\hat{\theta}^*_c = [8.252e6, 67.28, 1406.80, 0.1657]^T,$$

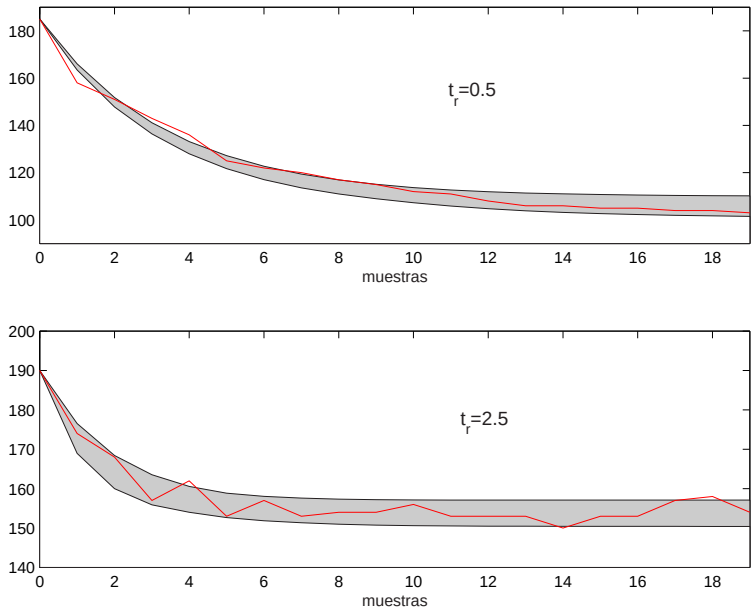

Figura 7.32: $y_{ide}(t)$ y la envolvente de los modelos FPS^*_{ide}.

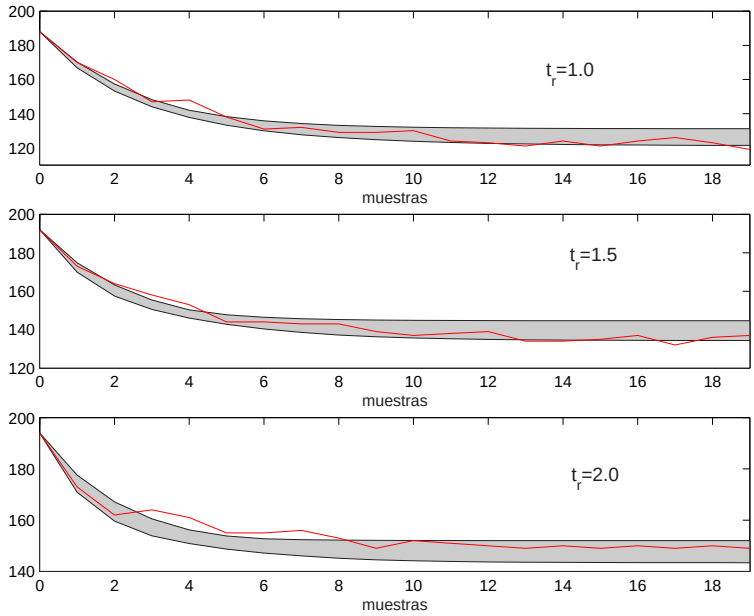

Figura 7.33: $y_{val}(t)$ y la envolvente de los modelos FPS^*_{ide}.

$$\hat{\theta}^*_{pi} = [8.493e6, 82.56, 11.38, 0.1142]^T.$$

Estos modelos producen los siguientes valores para las normas N_1, N_2, N_3 y N_4:

$$N_1(\hat{\theta}^*_c) = 6.74V/seg.,\ N_1(\hat{\theta}^*_{pi}) = 7.41V/seg.$$

$$N_2(\hat{\theta}^*_c) = 2.21V/seg.,\ N_2(\hat{\theta}^*_{pi}) = 2.37V/seg.$$

$$N_3(\hat{\theta}^*_c) = 8.83V/seg.,\ N_3(\hat{\theta}^*_{pi}) = 7.41V/seg.$$

$$N_4(\hat{\theta}^*_c) = 3.12V/seg.,\ N_4(\hat{\theta}^*_{pi}) = 2.24V/seg.$$

por lo tanto $\hat{\theta}^*_c$ no quedaría validado (pues $N_3(\hat{\theta}^*_c) > 8V/seg$ y además $N_4(\hat{\theta}^*_c) > 2.5V/seg.$), mientras que $\hat{\theta}^*_{pi}$ sí pues pertenecen a $FPS = FPS_{ide} \bigcap FPS_{val}$ y por lo tanto, su elección como modelo nominal resultaría más adecuada.

La figura 7.34 muestra la localización de los modelos $\hat{\theta}^*_c$, $\hat{\theta}^*_{pi}$ junto con el FPS^*_{ide} y los modelos de proyección restringida $\hat{\Theta}^*_{Pr}$.

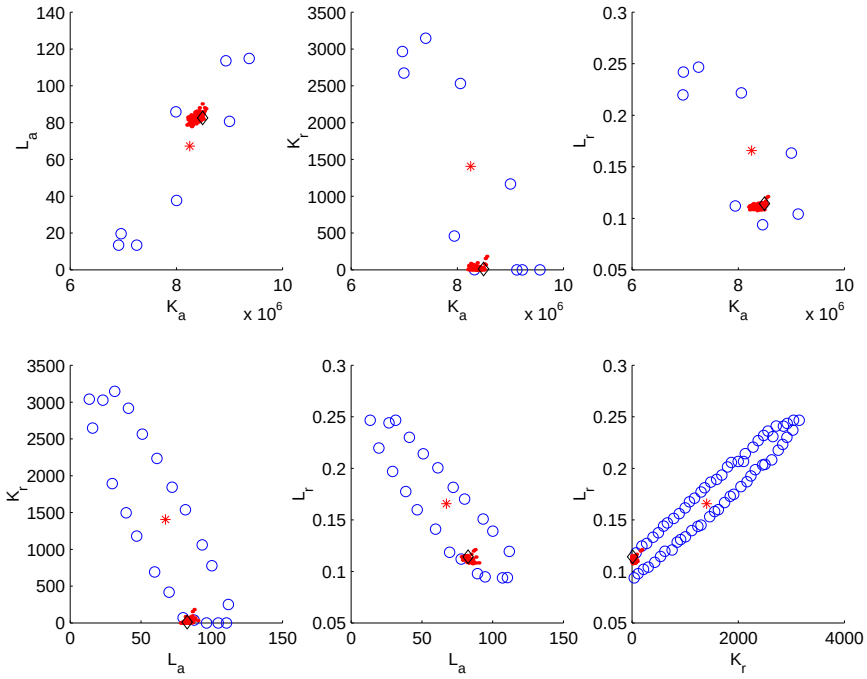

Figura 7.34: Modelos nominales óptimos. (*) Aproximación al centro de Chebyshev $\hat{\theta}^*_c$, ($\diamond$) modelo nominal de proyección interpolada restringida $\hat{\theta}^*_{pi}$ y ($\cdot$) modelos de proyección restringida $\hat{\Theta}^*_{Pr}$.

En el trabajo de Starmer [150] se identifica el siguiente modelo nominal mediante programación no lineal

$$\hat{\theta}_{Starmer} = [7.49e6, 43.19, 1.439, 0.1148]^T$$

sobre los datos $\{\Omega_{ide}, \Omega_{val}\}$, para minimizar

$$N_5(\theta) = \|\mathbf{e}(\theta, \{\Omega_{ide}, \Omega_{val}\})\|_2^{\mathbf{w}}, \;\; \mathbf{w}_i = 1/N_{\{\Omega_{ide},\Omega_{val}\}} \; \forall i \in [1, \ldots, N_{\{\Omega_{ide},\Omega_{val}\}}], \qquad (7.51)$$

con el siguiente resultado que se compara con el producido por $\hat{\theta}_c^*$ y $\hat{\theta}_{pi}^*$

$$N_5(\hat{\theta}_c^*) = 0.354V/seg., \; N_5(\hat{\theta}_{pi}^*) = 0.292V/seg., \; N_5(\hat{\theta}_{Starmer}) = 0.41V/seg.$$

Como se puede apreciar $\hat{\theta}_{Starmer}$ presenta un resultado peor comparado con $\hat{\theta}_c^*$ y $\hat{\theta}_{pi}^*$, aún cuando estos últimos no fueron estimados para minimizar $N_5(\theta)$[24]. Se ha comprobado que $\hat{\theta}_{Starmer}$ no responde al mínimo global. En [112] se obtiene un mínimo de $N_5(\theta)$[25] en

$$\hat{\theta}_{N_5} = [7.02e6, 663.577, 215.06, 0.1046]$$

$$N_5(\hat{\theta}_{N_5}) = 0.263V/seg., \; N_3(\hat{\theta}_{N_5}) = 10.125V/seg.$$

que como se puede comprobar presenta el mejor resultado respecto de N_5 aunque no pertenece al FPS pues para la norma N_3 el resultado supera la cota de $8V/seg$ establecida.

7.5. Conclusiones

A lo largo del capítulo, se ha presentado una metodología basada en los algoritmos $\not\epsilon$-MOGA y ϵ-GA, que permite caracterizar el conjunto de parámetros factibles FPS de modelos no lineales bajo incertidumbre paramétrica. El problema de IR se formula asumiendo la existencia de varias normas del error de identificación al mismo tiempo permitiendo así, por ejemplo, acotar el error a nivel de muestra ($||\cdot||_\infty$) o a nivel de experimento (p.e. $||\cdot||_1$ o $||\cdot||_2$) simultáneamente.

Las características de la metodología son:

- No establece ninguna restricción respecto tipo de modelo utilizado, siempre que sus salidas puedan ser obtenidas mediante simulación.
- Dado que las normas sobre el IR son tenidas en cuenta al mismo tiempo, se reduce el coste computacional, ya que, la intersección de los diferentes FPS_i se realiza de forma implícita.
- Permite caracterizar FPS no convexos e incluso inconexos.
- Al no aproximar el FPS mediante ortotopos, elipsoides, etc. se evita la conservatividad.
- Si crece la longitud del experimento (N), la complejidad computacional se preserva y el coste computacional sólo crece de forma proporcional a N.

[24] De igual modo $\hat{\theta}_c^*$ y $\hat{\theta}_{pi}^*$ presentan mejores resultados que $\hat{\theta}_{Starmer}$ sobre el resto de normas.
[25] Para su optimización se ha utilizado un GA clásico.

Se ha presentado un enfoque que ayuda a determinar las cotas η_i que deben de establecerse para las diferentes normas $N_i(\theta)$ a partir del análisis del frente de Pareto que se obtiene cuando se minimizan dichas normas simultáneamente. De esta manera, es posible escoger η_i para que $FPS \neq \emptyset$, y a través del procedimiento de validación (analizando las envolventes y/o el FPS^*) detectar una elección de η_i demasiado conservativa.

Adicionalmente, la técnica permite obtener una buena aproximación del modelo nominal de peor caso $\hat{\theta}_c^*$ calculando el centro de Chebyshev del FPS^*, como se ha podido comprobar.

Este modelo, al igual que la determinación del FPS es sensible a las cotas η_i, por ello, se ha planteado como alternativa a éste la determinación del modelo de proyección interpolada restringida $\hat{\theta}_{pi}^*$. Este modelo es óptimo en dos sentidos:

- Sobre el error de identificación, ya que pertenece al frente de Pareto, es decir, es un modelo de proyección.
- Sobre el error de estimación en el espacio de parámetros, ya que, es el modelo de proyección más cercano al centro de Chebyshev del FPS.

Los ejemplos de IR presentados con datos reales de un sistema térmico y del bloqueo que produce la cibenzolina sobre una célula cardíaca, han puesto de manifiesto la flexibilidad y potencia de la metodología de IR propuesta. En el caso del modelo de bloqueo de fármacos resulta evidente que el modelo nominal propuesto $\hat{\theta}_{pi}^*$ pertenece al FPS y es una buena solución de compromiso sobre las normas planteadas de forma simultánea.

Capítulo 8

Identificación Robusta del Modelo Climático de un Invernadero

8.1. Introducción

El control de los procesos bajo invernadero (control del clima, riego y fertilización) es requerido cada vez más como una exigencia de la agricultura de precisión para permitir aumentar tanto la calidad como la cantidad de los productos cultivados.

Tradicionalmente, el cultivo bajo invernadero se llevaba a cabo por personal que, a pie de invernadero, manejaba sistemas manuales de actuación o como mucho semiautomáticos. En invernaderos de mayor grado de tecnificación se usaban sistemas de control del tipo ON/OFF, PI y/o PID en bucles independientes. Generalmente, el sintonizado era manual mediante procedimientos de prueba y error y sin hacer uso de modelos matemáticos, lo que evidentemente no conducía a unas prestaciones óptimas.

En los últimos tiempos, empiezan a desarrollarse y aplicarse sistemas con estrategias de control más sofisticadas gracias al empleo de técnicas de modelado e identificación, todo ello apoyado en las nuevas tecnologías de la producción, de la información y las comunicaciones [165, 130].

La problemática del control de invernaderos es fuertemente dependiente de las áreas geográficas, las soluciones que son válidas en unas regiones deben adaptarse o cambiar para adecuarse a otras. En particular, en los países Mediterráneos, los altos índices de radiación y la elevada temperatura y humedad en la época estival constituyen un factor diferenciador respecto a otras regiones del Norte de Europa. Hasta ahora, una gran parte de los controladores diseñados para invernaderos se asocian a una única variable de control, la temperatura, dando lugar a controladores monovariables. Bajo las citadas condiciones estivales en regiones mediterráneas, ese control resulta del todo insuficiente y debe ser complementado con el control de la humedad [11], exigiendo que los controladores sean multivariables. Si se identifica de forma conjunta al control de estos aspectos (humedad y temperatura) como 'control climático' y desde el punto de vista del ingeniero de control, este bucle multivariable se puede controlar a través de actuadores como ventanas, nebulizadores, mallas de sombreo, etc., sin olvidar el sistemas de calefacción.

El proceso multivariable así definido es además de naturaleza no lineal e influyen en él procesos biológicos que complican de una manera notable el desarrollo de un modelo matemático. Ante un problema de esta envergadura surgen varias alternativas. Una de ellas es tratar el proceso como una caja negra sin utilizar información *a priori* y ajustar un modelo tipo red neuronal [146, 107] o conjunto borroso [48]. Un importante inconveniente, si se utiliza este tipo de técnicas, es precisamente la falta de relación física entre los parámetros del modelo y las magnitudes fundamentales del cultivo que los convierte en modelos poco entendibles. Otra alternativa consiste en modelar los fenómenos físicos y fisiológicos bien conocidos que se producen en un invernadero a partir de la formulación de ecuaciones de primeros principios basados en balances de masa y energía [22, 24]. En este caso los parámetros del modelo sí tienen un significado físico, pero el problema aparece cuando se intentan ajustar muchos de esos parámetros, lo cual resulta dificultoso ya que las discrepancias entre el modelo y el proceso suelen ser importantes.

La obtención de modelos fiables implica, por una parte, disponer de ecuaciones basadas

en primeros principios lo suficientemente representativas de los procesos que se llevan a cabo bajo invernaderos y por otra disponer de alguna técnica que permita ajustar los parámetros para reducir al máximo las discrepancias entre los datos reales procedentes de los procesos bajo invernadero y los que se obtendrían de los modelos propuestos.

Este será el objetivo principal de este capítulo, es decir, el modelado de un invernadero y la identificación de sus parámetros. Algunos de estos parámetros son función del tiempo y, por lo tanto, inciertos (por ejemplo aquellos relacionados con el cultivo) de ahí que resulte interesante determinar su incertidumbre a través de un proceso de identificación robusta.

Con un adecuado modelo es posible disponer de información muy relevante de cara al diseño del propio invernadero (dimensiones, actuadores, etc.), a la realización de predicciones a largo plazo con el fin de gestionar la producción, así como al control del invernadero, en particular usando estrategias basadas en modelos de predicción.

Los ingenieros agrónomos llevan tiempo perfeccionado modelos de procesos físicos y fisiológicos de invernaderos. En [149] se presenta un trabajo pionero en la descripción del modelo de humedad de un invernadero, basado en la obtención de un modelo no lineal de primeros principios de la humedad a través de la definición del balance de flujos de condensación, ventilación y transpiración. En este último caso se emplea la ecuación de Penman-Monteith [128] que incorpora las medidas del déficit de saturación y radiación para su evaluación, este modelo sigue estando de actualidad para el diseño de la ventilación en invernaderos [145]. El modelo de humedad se complementa con los modelos de balance energético a diferentes niveles. De nuevo, se construye una ecuación de primeros principios del balance de flujos térmicos asociados a la ventilación, convección, conducción y de calor latente debidos a la transpiración de la planta [11, 92] que definen la evolución de la temperatura. En función de los diferentes volúmenes del invernadero y el suelo es posible definir ecuaciones de evolución de la temperatura para cada uno de ellos y su interacción. Una mayor o menor complejidad del modelo es posible en función del número de volúmenes seleccionados que dan lugar a un mayor o menor número de ecuaciones diferenciales [20, 140].

En el resto del capítulo, se tratarán los siguientes aspectos: a lo largo de la sección 8.2 se abordará el modelado climático, basado en primeros principios, de un invernadero. En la sección 8.3 se llevará a cabo la identificación robusta (IR) del mismo utilizando la metodología presentada en el capítulo 7. Para terminar en la sección 8.4 se presentarán las conclusiones más importantes del capítulo.

8.2. Modelo del invernadero

A lo largo de esta sección se describirá el modelo climático no lineal del invernadero en espacio de estados a partir de los primeros principios.

El invernadero es considerado como un volumen de aire delimitado por las paredes, el techo y el suelo. El modelo en espacio de estados puede ser obtenido a partir de los

balances de masa y energía, incluyendo el comportamiento biológico de las plantas. Es posible establecer dos subsistemas, el volumen de aire y el suelo, actuando este último como una masa térmica [4]. Las variables de estado que describen el comportamiento climático son la temperatura $\hat{T}_i$ y humedad $\hat{H}R_i$ (o humedad absoluta H_i) en el aire y temperatura del suelo T_m (llamada temperatura de la masa térmica)[1].

El balance de masa de agua y energía en el aire establecen las dos primeras ecuaciones de estado.

$$\rho v_i \frac{dH_i}{dt} = F_v + C_{sat}(E + fog), \tag{8.1}$$

$$v_i \rho c_p \frac{d\hat{T}_i}{dt} = Q_s - Q_{cc} + Q_m - Q_v - C_{sat}(Q_e + Q_n) + W. \tag{8.2}$$

Donde la humedad absoluta interior H_i está en Kg_{H_2O}/Kg_{aire}, la temperatura interior $\hat{T}_i$ en ^oC, el volumen del invernadero v_i in m^3, la densidad del aire ρ en Kg_{aire}/m^3 y el calor especifico del aire c_p en $J/Kg\ ^oC$.

Los flujos en el balance de masa son (todos en Kg_{H_2O}/s)[2]:

- F_v: Flujo de renovación debido a la apertura de la ventana.
- E: Evapotranspiración del cultivo estimada a partir de la ecuación de Penman-Monteith [128] y tiene importantes no linealidades.
- fog: Aporte de agua del sistema de nebulización.

Los términos del balance de energía son (todos en W):

- Q_s: Energía solar suministrada al aire.
- Q_{cc}: Intercambio de energía por conducción y convección.
- Q_m: Intercambio de energía con la masa térmica.
- Q_e: Pérdidas de energía debidas a la evapotranspiración del cultivo.
- Q_n: Pérdidas de energía por nebulización.
- Q_v: Intercambio de energía debido a la ventilación.
- W: Energía aportada por la calefacción.

[1] Se utiliza $\hat{}$ para las variables de salida del modelo.
[2] C_{sat} es el coeficiente de saturación del aire, adimensional.

La tercera ecuación de estado se obtiene de un balance de energía sobre la masa térmica.

$$A_i C_m \frac{dT_m}{dt} = Q_{sm} - Q_m - Q_f. \tag{8.3}$$

Donde la temperatura de la masa térmica T_m está en ^oC y la capacidad calorífica de la masa térmica C_m en $J/m^2\ ^oC$.

Los términos del balance de energía son (todos en W/m^2):

- Q_m: Intercambio de energía entre la masa térmica y el aire interior.
- Q_{sm}: Energía almacenada por la masa térmica durante el día.
- Q_f: Pérdidas hacia el fondo del suelo.

Un diagrama del modelo, desde un punto de vista E/S, se muestra en la figura 8.1.

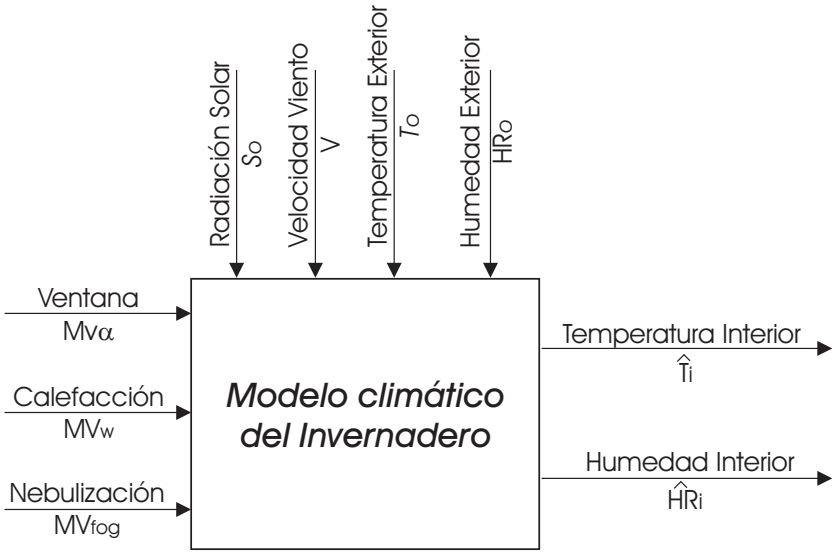

Figura 8.1: Modelo climático del invernadero.

Las variables de salida son:

- Humedad interior ($\hat{H}R_i$ en %).
- Temperatura interior ($\hat{T}_i$ en ^oC).

Las variables de entrada manipulables son:

- Control de la apertura de la ventana ($MV_\alpha \in [0, 100]\,\%$).
- Control de la calefacción ($MV_w \in [0, 100]\,\%$).

- Control de la nebulización ($MV_{fog} \in [0, 100]\,\%$).

Las perturbaciones medibles son:

- Radiación solar (S_o en $W \cdot m^{-2}$).
- Temperatura exterior (T_o en ^oC).
- Humedad exterior (HR_o en $\%$).
- Velocidad del viento (V en $m/seg.$).

Como puede verse en el apéndice B el modelo tiene una gran cantidad de parámetros. Algunos de ellos se pueden determinar fácilmente, por ejemplo, el volumen y el área del invernadero, pero hay otros como la conductancia estomática máxima que no lo son tanto. La complejidad del modelo y la gran cantidad de parámetros desconocidos, unido a que algunos de ellos varían con el tiempo, hacen que su ajuste no sea una tarea trivial.

8.3. Identificación Robusta del Invernadero

Una vez descrito el modelo del invernadero se procede a realizar la IR del mismo. Primero se verán los aspectos más importantes relacionados con la selección de los datos que se utilizarán en el proceso de identificación y validación. Después, una vez se determinen las normas a aplicar sobre el error de identificación (EI), se abordará la identificación de los modelos de proyección mediante la optimización multiobjetivo, con la intención de establecer las cotas a aplicar sobre cada norma. Seguidamente, tras seleccionar aquellos parámetros a los que se les asociará incertidumbre, se abordará la determinación del FPS^*_{ide} y los modelos nominales óptimos. Por último, se abordará la validación del FPS^*_{ide} y dichos modelos nominales.

8.3.1. Planificación de los experimentos

Una parte vital en el proceso de identificación es la planificación de los experimentos. Los aspectos que se analizarán en este apartado serán:

1. Selección de las señales de entrada y condiciones de funcionamiento del proceso.
2. Duración de los experimentos y frecuencia de muestreo de los datos.
3. Tratamiento de las señales.

Los aspectos del punto 1 son especialmente importantes cuando se habla de procesos cuyo comportamiento es claramente no lineal. En este caso, los fenómenos que se dan bajo invernadero además de complejos y no lineales son variantes con el tiempo. Por otra

parte las condiciones de funcionamiento vienen potencialmente impuestas por el efecto de las perturbaciones, principalmente por la radiación solar y la temperatura exterior. Estas perturbaciones siguen un comportamiento típico que se repite a diario y que depende en gran medida de la época del año[3].

Todo ello hace que no resulte posible dar con un conjunto único de parámetros θ que permita reproducir el comportamiento del invernadero durante todo un año completo, por ello se ajustará el modelo para cubrir la dinámica correspondiente a la época estival (sin distinguir entre periodos nocturnos y diurnos) por ser la época del año donde resulta más interesante controlar el clima del invernadero[4].

En cuanto al punto 2, los ensayos durarán un periodo múltiplo de 24 horas. Cuanto más días se utilicen más representativo será el modelo de la época en cuestión. Sin embargo, un número de días grande hace que las simulaciones sean costosas y el tiempo de ajuste del modelo se incremente considerablemente. Por ello, en este caso, por compromiso el número de días seleccionado ha sido dos (no consecutivos), utilizando como señales de entrada las que se utilizan en el funcionamiento diario del invernadero. El periodo de muestro utilizado ha sido de 15 segundos, más que suficiente para capturar la dinámica de los procesos bajo invernadero.

Para el proceso de identificación, se utilizarán los datos Ω_{ide} correspondientes a los días 11 y 15 de junio de 2002 y para la validación los datos Ω_{val1}, Ω_{val2}, Ω_{val3} y Ω_{val4} correspondientes a los días 20 de junio, 28 de julio, 22 de agosto y 8 de septiembre de 2002, respectivamente[5]. En el apéndice B puede verse la evolución temporal de todos los datos.

Relacionado con el punto 3, tratamiento de las señales, es necesario tratar la información que generan los sensores de radiación solar y velocidad del viento, pues son sensores que por su propia naturaleza física y por la de la magnitud que miden, introducen ruido que hay que eliminar para poder utilizar dicha información en el proceso de identificación de una manera adecuada.

Lo habitual es realizar un filtro de ventana o bien un filtro pasobajo, en este caso se ha utilizado, tanto para filtrar la radiación solar como la velocidad del viento un filtro digital pasobajo de primer orden como el que se muestra a continuación.

$$F(z^{-1}) = \frac{1-\alpha}{1-\alpha z^{-1}} \quad ; \quad \alpha = 0.95.$$

[3]La propia cubierta del invernadero se altera en verano, pintándola con cal, para reducir la radiación solar en su interior.

[4]En esta época no se utiliza la calefacción, por lo tanto este actuador no se tendrá en cuenta.

[5]Posiblemente lo ideal sería realizar la validación con todos los días de la época estival pero esto supondría un coste computacional muy elevado, de ahí que se hayan escogido algunos días teniendo en cuenta que: Estén repartidos a lo largo de la época estival, que los datos sean correctos (no existan interrupciones en la adquisición) y que reflejen situaciones diferentes en la evolución de las señales de entrada.

8.3.2. Identificación Multiobjetivo

Antes de abordar la identificación multiobjetivo que obtendrá el frente de Pareto a partir del cual se escogerán las cotas asociadas a las normas hay que tratar los siguientes aspectos:

- Adaptación del modelo.
- Selección de parámetros a identificar.
- Procedimiento para establecer las condiciones iniciales.

Relacionado con la adaptación del modelo, en el caso concreto del modelo climático del invernadero (figura 8.1) las ecuaciones de estado (8.1), (8.2) y (8.3) se adaptan de forma directa a las ecuaciones genéricas (6.1).

En cuanto a la selección de parámetros, para el caso del cultivo hidropónico de rosas bajo invernadero, el conjunto de parámetros candidato a ser estimado (θ) está asociado, por una parte, al cultivo específico del rosal, y por otra a parámetros asociados a distintas constantes de transmisión de calor, y temperaturas de referencia del invernadero[6]. Así pues, la adaptación del problema genérico en variables de estado al modelo del invernadero resulta:

$$\begin{aligned} \theta &= [gws_{max} \;\; gws_{min} \;\; k \;\; L \;\; gwb \;\; \tau \;\; a \;\; G_o \;\; Ac \\ &\quad\; C_m \;\; h_m \;\; T_{ref} \;\; \alpha_m \;\; k_a \;\; fog_{max}]^T, && (8.4) \\ \mathbf{u}(t) &= [MV_\alpha \;\; MV_{fog} \;\; S_o \;\; T_o \;\; HR_o \;\; V]^T, && (8.5) \\ \hat{\mathbf{y}}(t) &= \left[\hat{T}_i \;\; \hat{HR}_i\right]^T, && (8.6) \\ \mathbf{x}(t) &= \left[H_i \;\; \hat{T}_i \;\; T_m\right]^T. && (8.7) \end{aligned}$$

Para inicializar los estados se utilizará el mismo procedimiento que se utilizó en la IR del sistema térmico. Es decir, no se tendrá en cuenta el error de identificación durante las primeras $\varphi = 40$ muestras (correspondientes a 10 minutos ya que las muestras están distanciadas 15 segundos) y se fijarán los estados a partir de las medidas de las variables reales.

Por lo tanto, la primera variable de estado H_i puede ser inicializada directamente a partir del valor de las salidas $T_i(0)$ y $HR_i(0)$ en el instante inicial como se indica en el apéndice B (ecuaciones (B.7) y (B.8)). La segunda variable de estado $\hat{T}_i$ es al mismo tiempo una variable de salida, por lo tanto ésta puede ser inicializada con el valor que tome dicha salida en el instante inicial $T_i(0)$. La inicialización de la tercera variable de estado T_m, no resulta tan obvia dado que no se dispone de ningún sensor que la mida.

[6]En el Apéndice B se puede consultar el significado de cada parámetro objeto de identificación así como su rango de ajuste, los estudios analíticos previos han permitido aventurar estos rangos aproximados, lo que reduce la zona de búsqueda de una forma drástica.

Para estimar el valor inicial de $T_m(0)$ se utilizará la información de las entradas y salidas en el instante inicial y la ecuación (8.2) de balance de energía en el aire evaluada en dicho instante. Dado que las simulaciones se inician de noche (por lo tanto $S_o = 0$) y sin actuación de la nebulización (lo cual es lógico) la ecuación quedaría de la siguiente manera[7]:

$$v_i \rho c_p \frac{dT_i(0)}{dt} = -Q_{cc}(0) + Q_m(0) - Q_e(0) - Q_v(0).$$

que desarrollando quedaría de la siguiente manera:

$$\begin{aligned} v_i \rho c_p \frac{dT_i(0)}{dt} &= -A_i A_c (T_i(0) - T_o(0)) \\ &+ A_i h_m (T_m(0) - T_i(0)) - \lambda E(0) \\ &- \rho c_p A V(0)(a\alpha + G_o)(T_i(0) - T_o(0)) \end{aligned}$$

es posible asumir que $\frac{d\hat{T}_i(0)}{dt} = 0$ dado que su variación de noche es muy pequeña[8], de manera que $T_m(0)$ se obtendría de la siguiente manera:

$$\begin{aligned} T_m(0) = \frac{1}{A_i h_m} \Big(& A_i A_c (T_i(0) - T_o(0)) + \lambda E(0) \\ & + \rho c_p A V(0)(a\alpha + G_o)(T_i(0) - T_o(0)) \Big) + T_i(0), \end{aligned}$$

donde:

$$E(0) = \frac{A_i 2L\rho\, c_p\, D_i(0)\, gwb}{\left(\Delta + \gamma \left(1 + \frac{gwb}{gws_{min}}\right)\right) \lambda}.$$

Para determinar el FPS_{ide} asociado a los parámetros θ se van a plantear cuatro normas sobre el error de identificación. Las normas absoluta e infinito sobre la temperatura interior N_1 y N_3 y las normas absoluta e infinito sobre la humedad relativa interior N_2 y N_4, asumiendo así que, tanto la integral del error de identificación como el error en cada muestra estarán acotados, en ambas salidas, a lo largo del experimento.

$$N_1(\theta) = \|\mathbf{e}_1(\theta)\|_1^{\mathbf{w^1}},\ N_2(\theta) = \|\mathbf{e}_2(\theta)\|_1^{\mathbf{w^2}},\ N_3(\theta) = \|\mathbf{e}_1(\theta)\|_\infty^{\mathbf{w^3}},\ N_4(\theta) = \|\mathbf{e}_2(\theta)\|_\infty^{\mathbf{w^4}}. \quad (8.8)$$

donde $e_1(t_j, \theta) = T_i(t_j) - \hat{T}_i(t_j, \theta)$, $e_2(t_j, \theta) = HR_i(t_j) - \hat{H}R_i(t_j, \theta)$ y:

$$\mathbf{w}_j^1 = 0\ \forall j \in [1, \ldots, \varphi],\ \mathbf{w}_j^1 = \frac{1}{(N - \varphi)}\ \forall j \in [\varphi + 1, \ldots, N],$$

$$\mathbf{w}_j^2 = k_j \mathbf{w}_j^1,$$

[7]No tener en cuenta estas hipótesis simplemente repercutiría en una expresión más extensa a la hora de determinar $T_m(0)$.

[8]Para los días de verano que se van a utilizar en la identificación $\frac{dT_i(0)}{dt} \approx 0.1 \cdot 10^{-3} (^oC/seg.)$.

$$\mathbf{w}_j^3 = 0 \; \forall j \in [1, \ldots, \varphi], \; \mathbf{w}_j^3 = 1 \; \forall j \in [\varphi + 1, \ldots, N],$$
$$\mathbf{w}_j^4 = k_j \mathbf{w}_j^3,$$

con

$$k_j = \begin{cases} -0.02 * \hat{H}R_i(t_j) + 2.2 & si \quad \hat{H}R_i(t_j) \in [60 \ldots 100]\,\% \\ \\ 1 & si \quad \hat{H}R_i(t_j) \in [0 \ldots 60[\,\% \end{cases} . \tag{8.9}$$

Con el uso del parámetro k_j lo que se pretende es que los errores que se produzcan para humedades más elevadas se vean ponderadas dándoles una importancia relativa menor. El motivo está relacionado con el sensor de humedad comercial que se utiliza, ya que presenta una pérdida importante de precisión para estos valores (debido a la condensación que se suele producir sobre el sensor por falta de ventilación). La figura 8.2 muestra la relación entre k_j y $\hat{H}R_i(t_j)$ donde, se observa que los errores producidos para humedades relativas del orden del 100 % están ponderados por 0.2, es decir tiene un importancia relativa menor que los producidos para humedades del 60 %.

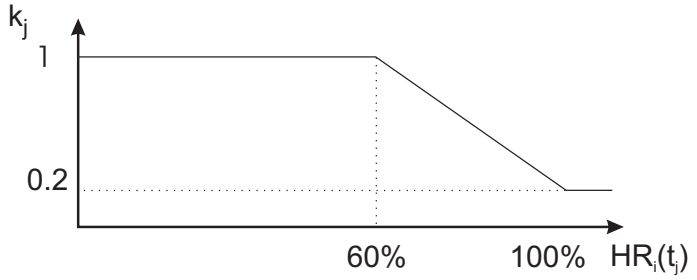

Figura 8.2: $k_j = f(\hat{H}R_i(t_j))$.

A continuación, se plantea el siguiente problema de optimización multiobjetivo

$$\min_{\theta \in D} \mathbf{J}(\theta) = \{N_1, N_2, N_3, N_4\}. \tag{8.10}$$

De la información que genere el frente de Pareto obtenido con el algoritmo $\not\epsilon^2$MOGA se escogerán las cotas η_1, η_2, η_3 y η_4 asociadas a cada norma.

Los parámetros del algoritmo $\not\epsilon^2$MOGA escogidos son los siguientes[9]:

- El espacio de búsqueda D viene determinado por los rangos establecidos para cada uno de los 15 parámetros del vector θ, que pueden consultarse en el apéndice B.
- $t_{max} = 60000$ y $n\_box_i = 70 \; \forall i \in [1 \ldots 4]$.

La figura 8.3 muestra el frente de Pareto correspondiente a los modelos óptimos de proyección $\hat{\mathbf{\Theta}}_P^*$ obtenidos como solución al problema de optimización multiobjetivo planteado.

[9]El resto de parámetros del algoritmo son los mismos que se utilizaron en la optimización de los MOP1...MOP5 del capítulo 3.

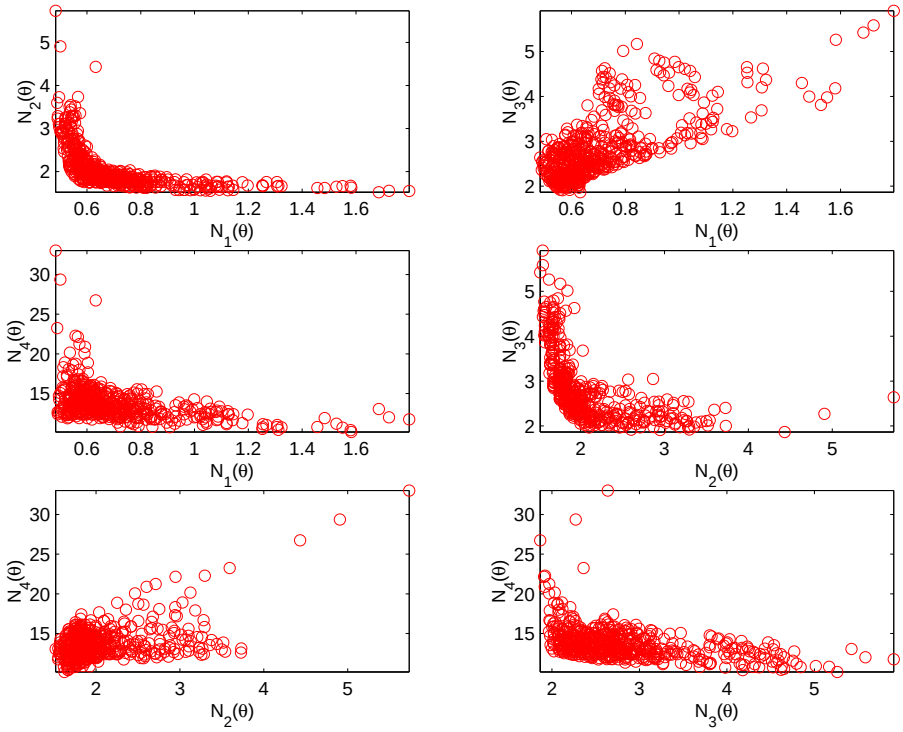

Figura 8.3: Proyecciones del frente de Pareto $\mathbf{J}(\hat{\mathbf{\Theta}}_P^*)$ sobre los planos $(N_1(\theta), N_2(\theta))$, $(N_1(\theta), N_3(\theta))$, $(N_1(\theta), N_4(\theta))$, $(N_2(\theta), N_3(\theta))$, $(N_2(\theta), N_4(\theta))$ y $(N_3(\theta), N_4(\theta))$.

De análisis del frente de Pareto se puede obtener el punto utópico $\mathbf{J}^{ideal}$ a partir de las siguientes cotas mínimas:

$$\eta_1^{min} = \text{mín}\, N_1(\theta) = 0.483^oC,\ \eta_2^{min} = \text{mín}\, N_2(\theta) = 1.51\,\%,$$
$$\eta_3^{min} = \text{mín}\, N_3(\theta) = 1.86^oC,\ \eta_4^{min} = \text{mín}\, N_4(\theta) = 10.13\,\%.$$

Con lo cual

$$\mathbf{J}^{ideal} = \{\eta_1^{min}, \eta_2^{min}, \eta_3^{min}, \eta_4^{min}\} = \{0.483, 1.51, 1.86, 10.13\},$$

y un posible modelo ideal podría obtenerse como:

$$\begin{aligned}\theta^{ideal} &= \arg\min_{\theta\in\hat{\mathbf{\Theta}}_P^*} ||\mathbf{J}(\theta) - \mathbf{J}^{ideal}||_2 = \\ &= [0.010078, 0.002865, 0.41594, 0.72374, 0.016825, \\ &\quad 0.30016, 0.0011995, 0.00025871, 19.519, 1.8596e^5, \\ &\quad 9.4531, 18.732, 0.012533, 1.4839, 0.0019266], \qquad (8.11)\end{aligned}$$

$$\mathbf{J}(\theta^{ideal}) = \{0.623, 2.16, 2.12, 12.74\}. \qquad (8.12)$$

Por lo tanto, si se escogen las cotas para que las predicciones realizadas por los modelos del FPS_{ide} no generen errores mayores de 3^oC en temperatura y $20\,\%$ en humedad como máximo y de media no mayores a 0.8^oC y $3\,\%$, es decir $\eta_1 = 0.8$, $\eta_2 = 3$, $\eta_3 = 3$ y $\eta_4 = 20$, se consigue un $\hat{\Theta}_{Pr} \neq \emptyset$ y por lo tanto un $FPS \neq \emptyset$.

8.3.3. Identificación Robusta

El FPS^*_{ide} no se va a asociar a los 15 parámetros de θ ya que, hay algunos que presentan una sensibilidad bastante baja. Por lo tanto, para seleccionar aquellos parámetros sobre los que se va a determinar su incertidumbre, se realiza el siguiente análisis de sensibilidad. Tomando como punto de referencia θ^{ideal} se modifica, uno a uno, cada parámetro a lo largo de su espacio de búsqueda y se evalúa la variación que experimenta cada norma. La tabla 8.1 muestra el resultado de dicho análisis de sensibilidad.

θ_i	$\frac{\Delta N_1}{\Delta \theta_i}$	$\frac{\Delta N_2}{\Delta \theta_i}$	$\frac{\Delta N_3}{\Delta \theta_i}$	$\frac{\Delta N_4}{\Delta \theta_i}$
gws_{max}	0.12	0.2	0.6	6
gws_{min}	0.15	4	0.8	8
k	0.04	0.8	0.2	3
L	0.8	6	1.5	22
gwb	0.3	6	1.3	10
τ	3	5	10	13
a	0.5	5	1.7	13
G_0	0.2	3	0.8	5
A_c	1	1	3.8	3
C_m	0.05	0.4	0.3	0.5
h_m	0.4	0.6	3	4
T_{ref}	0.004	0.05	0.05	0.2
α_m	0.003	0.04	0.01	0.1
k_a	0.02	0.07	0.06	0.1
fog_{max}	0.6	4	3	30

Tabla 8.1: Variación que experimentan las normas N_1, N_2, N_3 y N_4 cuando se modifica, respecto de θ^{ideal}, cada uno de sus parámetros de forma independiente a lo largo de su rango de búsqueda.

Como se puede observar en la tabla, los parámetros gws_{max}, gws_{min}, k, G_o, C_m, T_{ref}, α_m y k_a presentan una sensibilidad menor comparada con el resto y no serán considerados a la hora de determinar el FPS^*_{ide}, es decir, no será determinada su incertidumbre fijando su valor al que presenta el modelo ideal θ^{ideal}. De esta manera, se consigue reducir el espacio de búsqueda del FPS^*_{ide} y al mismo tiempo se reducirá considerablemente el número de modelos que lo compongan.

A continuación, se procede a determinar el FPS^*_{ide} utilizando el algoritmo ϵ-GA con la siguiente configuración:

- El espacio de búsqueda D viene determinado por los rangos establecidos para cada uno de los 7 parámetros que ahora componen el vector $\theta = [L, gwb, \tau, a, A_c, h_m, fog_{max}]$[10]. Estos rangos pueden consultarse en el apéndice B.

[10]Algunos de estos parámetros varían con el tiempo. Por ejemplo, el índice foliar L (estado del cultivo),

- $t_{max} = 40000$ y $\epsilon = [0.19, 0.0098, 0.08, 0.00099, 2.8, 5.8, 0.00398]$ de manera que el *grid* tenga 10 divisiones en los parámetros L, τ, a y A_c (los que presentan mayor sensibilidad) y 5 para gwb, h_m y fog_{max}.

La figura 8.4 muestra el resultado del proceso de optimización con ϵ-GA, es decir, el FPS^*_{ide}[11]. El algoritmo ha caracterizado el FPS_{ide} mediante 4208 modelos y la media de $J'(\partial FPS^*_{ide})$ es de 0.0019 lo que demuestra una buena convergencia del algoritmo (la media de $J'(\partial FPS^*_{ide})$ ideal sería 0). El número de evaluaciones de la función $J'(\theta)$ ha sido de 160100, es decir, aproximadamente la octava parte de las que hubiesen hecho falta si se hubiese evaluado $J'(\theta)$ en cada *box* del *grid*[12] (búsqueda exhaustiva).

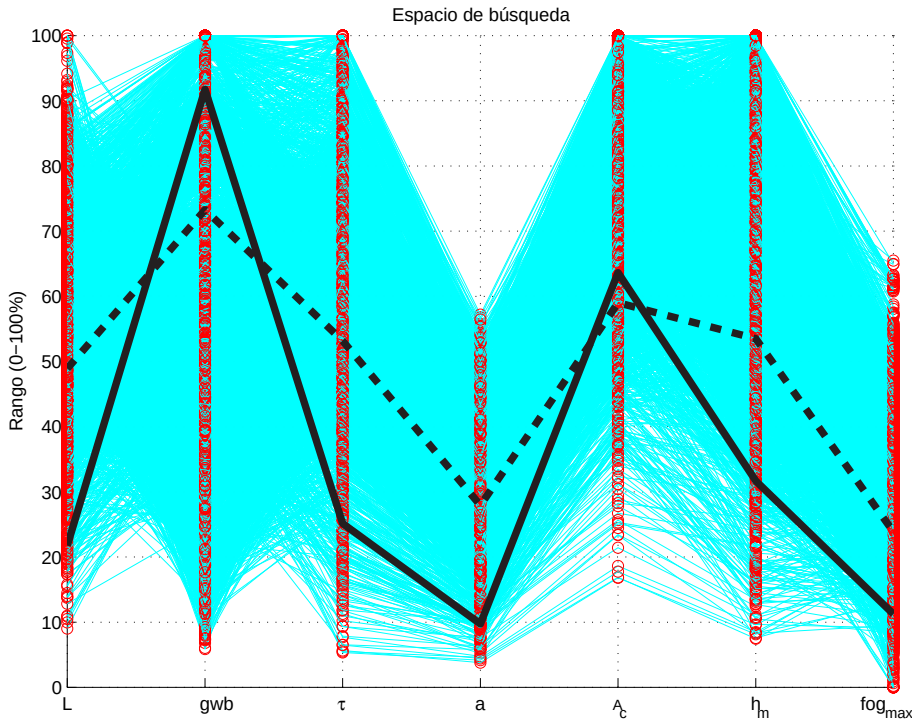

Figura 8.4: Mediante trazas se representa los modelos del FPS^*_{ide} dentro del espacio de búsqueda. Sobre el eje de abscisas las 7 dimensiones que componen el espacio de búsqueda. En ordenadas el valor de cada parámetro en % sobre su rango de búsqueda. Con trazo grueso aparecen reflejados los modelos nominales $\hat{\theta}^*_c$ (trazo discontinuo) y $\hat{\theta}^*_{pi}$ (trazo continuo).

La figura 8.5 muestra los datos Ω_{ide}, junto con la envolvente producida por el FPS^*_{ide}.

el coeficiente de transmisión de la cubierta del invernadero τ (suciedad en la cubierta, además se altera cuando se pinta la cubierta con cal para reducir la incidencia del sol), la nebulización máxima fog_{max} (que depende del estado del sistema de microaspersión que se ve alterado por la cal que se deposita en las boquillas), etc.

[11] No se presentan las proyecciones del FPS^*_{ide} por el elevado número de éstas que serían necesarias.

[12] Si se tiene en cuenta que evaluar 160100 veces la función $J'(\theta)$ supone, en tiempo, unas 4h utilizando una plataforma con 4 computadoras, si se hubiese utilizado búsqueda exhaustiva el tiempo hubiese sido de 32h aproximadamente y además la caracterización del FPS_{ide} hubiese sido peor en su contorno.

Se puede ver que la envolvente prácticamente consigue envolver los datos $Y_{ide}(t)$.

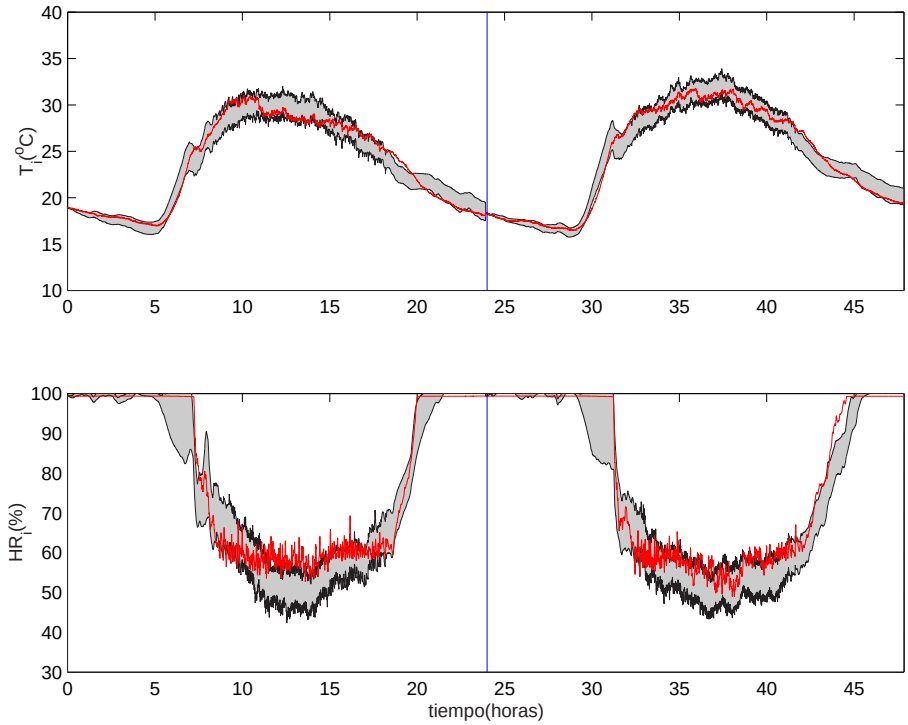

Figura 8.5: $Y_{ide}(t)$ y la envolvente producida por los modelos del FPS^*_{ide}.

A partir del FPS^*_{ide} se pueden obtener los modelos de peor caso $\hat{\theta}^*_c$ [13] y de proyección interpolada restringida[14] que serían los siguientes:

$$\hat{\theta}^*_c = [1.029, 0.0369, 0.525, 0.00286, 18.527, 16.518, 0.00484]^T,$$
$$\hat{\theta}^*_{pi} = [0.513, 0.0459, 0.301, 0.00106, 19.821, 10.184, 0.00233]^T.$$

Estos modelos, representados en la figura 8.4, producen los siguientes valores para las normas N_1, N_2, N_3 y N_4:

$$N_1(\hat{\theta}^*_c) = 0.767^oC,\ N_1(\hat{\theta}^*_{pi}) = 0.634^oC,$$

$$N_2(\hat{\theta}^*_c) = 2.703\,\%,\ N_2(\hat{\theta}^*_{pi}) = 2.170\,\%,$$

$$N_3(\hat{\theta}^*_c) = 2.427^oC,\ N_3(\hat{\theta}^*_{pi}) = 2.007^oC,$$

$$N_4(\hat{\theta}^*_c) = 15.161\,\%,\ N_4(\hat{\theta}^*_{pi}) = 13.916\,\%.$$

[13] Para su obtención se ha utilizado un GA clásico.

[14] Para su obtención ha sido necesario volver a determinar los nuevos modelos de proyección teniendo en cuenta que el espacio de búsqueda es el correspondiente a los 7 parámetros a los que se asocia el FPS^*_{ide}.

8.3.4. Validación

El criterio que se va a utilizar para la validación es el que determina si en el FPS_{ide} hay datos consistentes con Ω_{val} y las cotas $\eta_1 = 0.8$, $\eta_2 = 3$, $\eta_3 = 3$ y $\eta_4 = 20$ establecidas, es decir, $FPS \neq \emptyset$. En este caso en el FPS^*_{ide} hay 16 modelos consistentes con Ω_{val1}, 5 con Ω_{val2}, 51 con Ω_{val3} y 231 con Ω_{val4} con lo cual el FPS^*_{ide} quedaría validado. Las figuras 8.6 y 8.7 muestran los datos $\Omega_{val1} \ldots \Omega_{val4}$, junto con la envolvente producida por el FPS_{ide}.

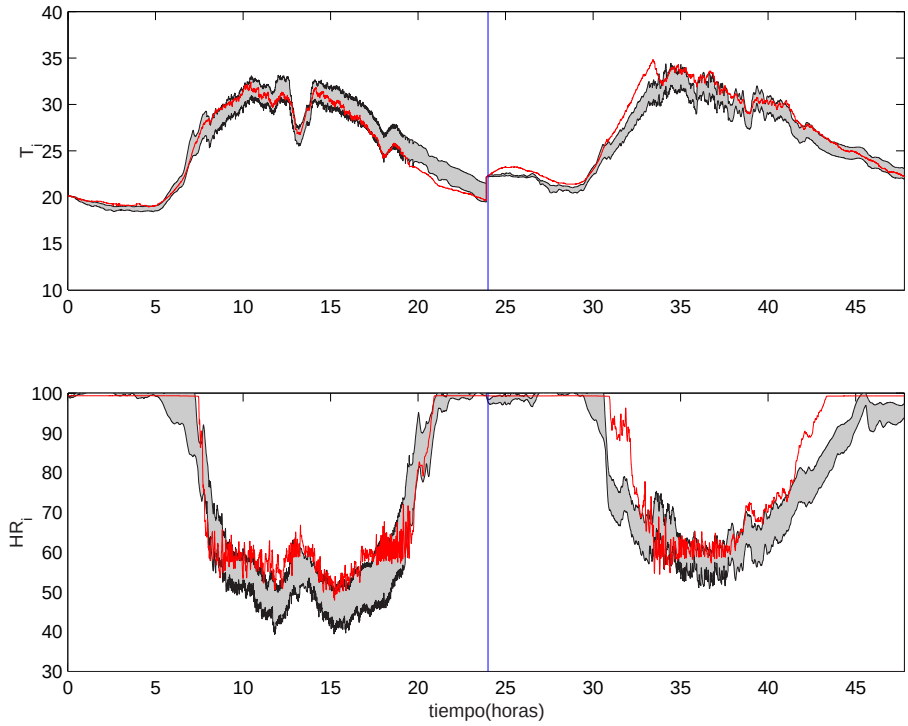

Figura 8.6: $Y_{val1}(t)$ e $Y_{val2}(t)$ y la envolvente producida por los de modelos del FPS^*_{ide}.

En cuanto a los modelos nominales se refiere, éstos producen los valores que se muestran en la tabla 8.2 para las normas N_1, N_2, N_3 y N_4 con los datos de validación $\Omega_{val1} \ldots \Omega_{val4}$. Del análisis de estos resultados se puede deducir que, aunque estrictamente ninguno de los dos modelos quedaría validado, producen resultados aceptables. Especialmente el modelo $\hat{\theta}^*_{pi}$ que resultaría validado para Ω_{val3} y Ω_{val4} y sólo violaría, levemente, tres cotas:

$$N_2(\hat{\theta}^*_{pi})(\,\%) = 3.195 > \eta_2 = 3\,\% \Rightarrow \Omega_{val1},$$

$$N_1(\hat{\theta}^*_{pi})(^oC) = 0.895 > \eta_1 = 0.8\,\% \Rightarrow \Omega_{val2},$$

y

$$N_3(\hat{\theta}^*_{pi})(^oC) = 3.828 > \eta_3 = 3\,\% \Rightarrow \Omega_{val2}.$$

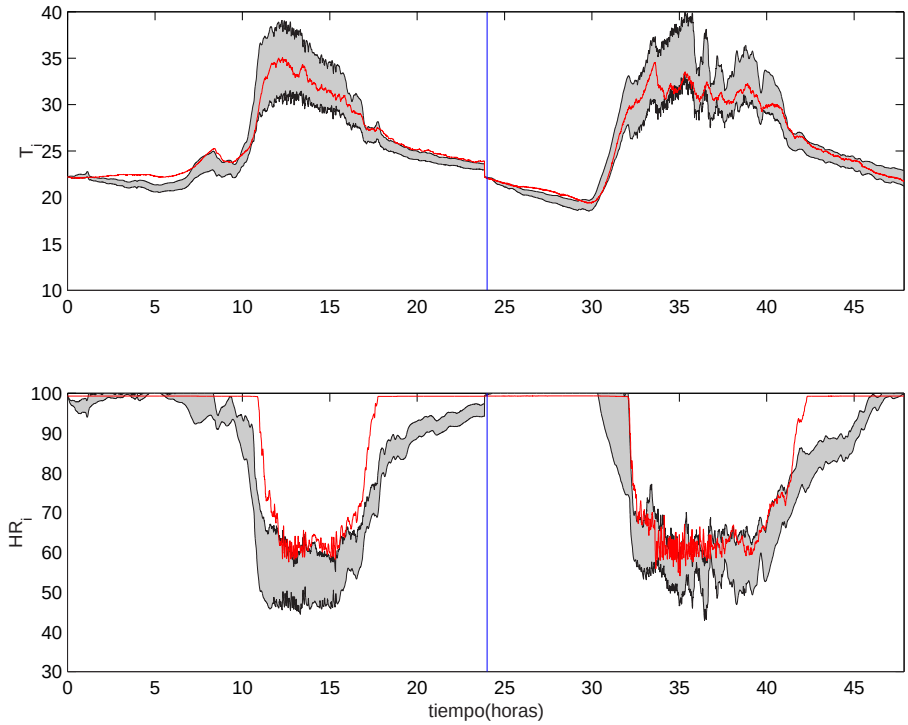

Figura 8.7: $Y_{val3}(t)$ e $Y_{val4}(t)$ y la envolvente producida por los modelos del FPS^*_{ide}.

	Ω_{val1}	Ω_{val2}	Ω_{val3}	Ω_{val4}
$N_1(\hat{\theta}^*_c)(^oC)$	**0.732**	0.97	1.060	0.6985
$N_1(\hat{\theta}^*_{pi})(^oC)$	0.763	**0.895**	**0.726**	**0.619**
$N_2(\hat{\theta}^*_c)(\,\%)$	3.840	2.752	**2.556**	2.1343
$N_2(\hat{\theta}^*_{pi})(\,\%)$	**3.195**	**2.490**	2.850	**1.994**
$N_3(\hat{\theta}^*_c)(^oC)$	2.319	4.188	3.455	3.550
$N_3(\hat{\theta}^*_{pi})(^oC)$	**2.202**	**3.828**	**2.042**	**2.800**
$N_4(\hat{\theta}^*_c)(\,\%)$	18.994	**12.792**	**11.612**	13.664
$N_4(\hat{\theta}^*_{pi})(\,\%)$	**18.785**	15.476	12.175	**11.993**

Tabla 8.2: Valores que producen los modelos $\hat{\theta}^*_c$ y $\hat{\theta}^*_{pi}$ para las normas N_1, N_2, N_3 y N_4 con los datos de validación $\Omega_{val1} \ldots \Omega_{val4}$. En negrita se destaca el modelo que produce mejor resultado para cada norma y datos. Aparecen subrayados los valores que superan las cotas establecidas para cada norma.

8.4. Conclusiones

En este capítulo se ha presentado la identificación robusta del modelo climático (temperatura y humedad) de un invernadero utilizando la metodología de IR presentada en esta tesis. El modelo no lineal del invernadero, que presenta 15 parámetros desconocidos, está basado en los balances de masa y energía que aparecen al considerar el invernadero como un volumen de aire delimitado por las paredes, el techo y el suelo e incluye el comportamiento biológico de las plantas. El objetivo ha sido ajustar el modelo para reflejar el comportamiento de la época estival por el interés que supone el control del clima en esta época.

Para abordar la IR se han planteado 4 normas sobre el EI, el error máximo y medio sobre la temperatura interior del invernadero y el error máximo y medio sobre la humedad relativa ponderando ésta última, de manera que las humedades elevadas tengan una incidencia relativa menor, debido a que el sensor de humedad no es fiable debido al fenómeno de condensación.

Se ha podido comprobar que algunos de los 15 parámetros presentan una nivel de sensibilidad bajo, es decir, su variación dentro del rango de búsqueda no tiene una incidencia elevada sobre las normas del EI, de ahí que se haya decidido seleccionar sólo 7 de los parámetros (los de mayor sensibilidad) para determinar el FPS^*_{ide} asociado a éstos. El resto de parámetros se han establecido a partir del modelo ideal θ_{ideal}, que es el modelo de proyección interpolada que minimiza la distancia al punto utópico $\mathbf{J}^{ideal}$, y que ha sido determinado a partir del frente de Pareto utilizado, además, para establecer las cotas sobre las 4 normas del EI.

Para abordar la IR se han tomado los datos correspondientes a dos días del inicio del periodo estival y se ha caracterizado el FPS^*_{ide} mediante 4208 modelos que han conseguido capturar la dinámica del invernadero con bastante verosimilitud, como se pone de manifiesto mediante la evolvente producida por el FPS^*_{ide} que, de forma bastante ajustada, prácticamente envuelve los datos $Y_{ide}(t)$. Para la validación se han utilizado los datos de 4 días repartidos a lo largo de la época estival. El FPS^*_{ide} ha sido validado para los 4 días de forma independiente, ya que contiene modelos consistentes con las normas y cotas establecidas. La envolvente producida por el FPS^*_{ide} para los datos de validación, aunque sí que reproduce bien el comportamiento del invernadero, no envuelte los datos, lo que pone de manifiesto la existencia de dinámica no modelado considerable, como se esperaba.

En cuanto a los modelos nominales, ninguno de ellos ha sido validado, de hecho en el FPS^*_{ide} no hay ningún modelo que valide todos los datos Ω_{val1}, Ω_{val1}, Ω_{val1} y Ω_{val1} al mismo tiempo. Por otra parte, de entre los dos modelos nominales se puede ver que $\hat{\theta}^*_{pi}$ presenta mejores resultados, en cómputo global, que $\hat{\theta}^*_c$. Para conseguir un modelo nominal que resultase validado se podría: Modificar el modelo para que cubriese mejor la dinámica del invernadero y/o aumentar las cotas asociadas a las normas.

Capítulo 9

Conclusiones de la Tesis y Trabajos Futuros

9.1. Conclusiones principales y aportaciones

En esta tesis se ha presentado una metodología de identificación robusta paramétrica para determinar el conjunto de parámetros factibles FPS (parámetros del modelo nominal y la incertidumbre en los mismos) que aparece cuando se asume un enfoque determinístico (error de identificación desconocido pero acotado). La metodología determina el FPS y el modelo nominal con la ayuda de dos algoritmos evolutivos, ϵ-GA desarrollado específicamente para optimizar funciones multimodales con infinitos mínimos globales y $\not\epsilon$MOGA que permite obtener el frente de Pareto asociado a un problema de optimización multiobjetivo. Las principales características de la metodología es la potencia y flexibilidad de la misma, ya que permite:

- Utilizar cualquier tipo de modelo paramétrico para el proceso, siempre y cuando sus salidas puedan calcularse mediante simulación.
- Acotar el error de identificación mediante diferentes normas simultáneamente pudiendo, de esta manera, establecer diferentes consideraciones/características, al mismo tiempo, que deben cumplir los modelos del FPS.
- Ser no conservativo ya que el FPS no se aproximará mediante ortotopos, elipsoides, etc.
- Caracterizar FPS de cualquier tipo: convexos, no convexos e inconexos.
- Determinar las cotas asociadas a cada norma, de tal manera que se asegure un $FPS \neq \emptyset$.
- Calcular, fácilmente, una buena aproximación al modelo nominal de peor caso $\hat{\theta}_c^*$ y el modelo $\hat{\theta}_{pi}^*$ óptimo en términos del error de identificación y del error de estimación en el espacio de los parámetros.

Con estas características, la metodología resulta especialmente útil en la IR de modelos no lineales de procesos de primeros principios, como los presentados en esta tesis a modo de ejemplo.

De forma detallada las principales conclusiones/aportaciones de esta tesis son:

1. Estudio detallado de los aspectos básicos de los algoritmos evolutivos: tipo de codificación, operadores y ajuste de parámetros del propio EA y aspectos particulares: tratamiento de restricciones, implementación paralela y mecanismos específicos para la optimización de funciones multimodales. Dentro de este campo el autor de la tesis ha realizado trabajos relacionados con la utilización de algoritmos genéticos en identificación no lineal, diseño de PIDs y control predictivo no lineal [21, 71, 76, 75, 79].
2. Estudio de los métodos clásicos de resolución de problemas de optimización multiobjetivo y de los algoritmos evolutivos más importantes utilizados en la resolución de MOPs clasificándolos como de primera y segunda generación.

3. Desarrollo del algoritmo evolutivo de optimización multiobjetivo ϵ-MOGA, basado en el algoritmo ϵ-MOEA presentado en [45], con las aportaciones de incorporar operadores de cruce (recombinación intermedia extendida) y mutación (aleatoria con distribución *gaussiana*) con ajuste de parámetros no adaptativo basado en funciones y utilizando una arquitectura paralela maestro-esclavo. ϵ-MOGA es un MOEA de segunda generación (elitista) basado en el concepto ϵ-dominante que permite obtener un conjunto de soluciones que, converge de forma distribuida hacia el frente de Pareto, utilizando recursos de memoria limitados y sin generar deterioro como se ha demostrado.

4. Evaluación del algoritmo ϵ-MOGA utilizando los problemas MOP1...MOP5 y comparación con ϵ-MOEA y los algoritmos de búsqueda aleatoria y exhaustiva poniendo de manifiesto que ϵ-MOGA presenta mejores prestaciones en cuanto a convergencia y diversidad de soluciones encontradas sobre el frente de Pareto.

5. Desarrollo del algoritmo evolutivo multiobjetivo $\not\epsilon$-MOGA como una variante original del algoritmo ϵ-MOGA que permite ajustar dinámicamente los límites del frente de Pareto capturando sus extremos. Esta ventaja respecto al algoritmo ϵ-MOGA conlleva la desventaja de no poder asegurar no deterioro de forma teórica, aunque sí que se han demostrado sus prestaciones mediante la resolución de los problemas MOP1...MOP5 y la de un problema de ingeniería donde se determinan las secciones de tres barras de una estructura mecánica minimizando simultáneamente el volumen total de las mismas y una combinación lineal de los desplazamientos de un punto de dicha estructura.

6. Desarrollo del algoritmo evolutivo ϵ-GA diseñado específicamente para optimizar funciones mono objetivo multimodales que presentan múltiples (incluso infinitos) óptimos globales. Está inspirado en el algoritmo ϵ-MOGA del que hereda su estructura, operadores de cruce y mutación, arquitectura paralela y, añade un mecanismo de reparación para evitar la convergencia prematura del algoritmo. Utiliza los conceptos de quasi mínimo global y box-representante presentados en esta tesis para demostrar que la solución converge de forma distribuida hacía el conjunto de soluciones óptimas globales utilizando recursos de memoria limitados.

7. Evaluación del algoritmo ϵ-GA utilizando los problemas OP1...OP5 diseñados en la tesis para tal propósito y comparación con dos estrategias basadas en el algoritmo SQP (*multi-start* SQP). De la comparación se ha podido demostrar que ϵ-GA consigue obtener mejores prestaciones en cuanto a número de soluciones quasi globales obtenidas se refiere y con una mejor distribución sobre las soluciones óptimas globales.

8. Estudio de los aspectos más importantes relacionados con la identificación paramétrica de sistemas, en especial sobre el criterio de optimalidad a utilizar, y la necesidad de utilizar optimizadores globales (por ejemplo, EAs) para abordar los problemas de optimización (no convexos, multimodales, etc.) que pueden resultar cuando se utilizan modelos no lineales y/o criterio de optimalidad cualesquiera.

9. Estudio de los conceptos básicos y formulación asociada a la identificación robusta cuando se utiliza un enfoque determinístico el cual conlleva asociada la determinación del FPS. Análisis de diferentes métodos para su determinación cuando se utilizan modelos no lineales respecto de sus parámetros, lo que podría trascender en FPS no convexos y/o inconexos dificultando su obtención.

10. Presentación de una metodología potente, flexible y computacionalmente eficiente, basada en los algoritmos $\not\epsilon$MOGA y ϵ-GA que permite caracterizar el conjunto de parámetros factibles FPS cuando se plantea un problema de identificación robusta paramétrica en modelos no lineales. Esta metodología ha supuesto:

    - El desarrollo de las funciones multimodales J y J' de manera que sus mínimos globales constituyan el ∂FPS o el FPS respectivamente, cuando se acota el EI mediante varias normas simultáneamente. Al tener en cuenta todas las normas al mismo tiempo se reduce el coste computacional, ya que, la intersección de los diferentes FPS_i asociados a cada norma $N_i(\theta)$ y cota η_i, se realiza de forma implícita.
    - La presentación de un enfoque que ayuda a determinar las cotas η_i, que deben de establecerse para las diferentes normas $N_i(\theta)$, a partir del análisis del frente de Pareto que se obtiene cuando se minimizan dichas normas simultáneamente. De esta manera, es posible escoger η_i para que $FPS \neq \emptyset$.

11. Determinación de un modelo nominal nuevo, el modelo de proyección interpolada restringida $\hat{\theta}^*_{pi}$, como alternativa al modelo nominal de peor caso $\hat{\theta}^*_c$. Este modelo resulta óptimo desde el punto de vista del error de identificación, ya que pertenece al frente de Pareto, es decir, es un modelo de proyección y, desde el punto de vista del error de estimación en el espacio de parámetros, ya que, es el modelo de proyección más cercano al centro de Chebyshev del FPS y, por supuesto, pertenece al FPS. A través de los ejemplos de IR presentados se ha puesto de manifiesto el buen comportamiento presentado por $\hat{\theta}^*_{pi}$ frente a $\hat{\theta}^*_c$, ya que resulta una buena solución de compromiso sobre las normas $N_i(\theta)$ planteadas de forma simultánea.

12. La identificación y validación robusta, con datos reales, de los parámetros de dos modelos no lineales, uno de un sistema térmico y otro que refleja el bloqueo que produce la cibenzolina sobre una célula cardíaca. Ambos ejemplos han servido para poner de manifiesto la potencia y flexibilidad de la metodología de IR propuesta. Parte de los resultados de la IR del sistema térmico han sido presentados en [77, 72, 73, 74, 78].

13. El modelado, identificación y validación del modelo climático (temperatura y humedad) de un invernadero, con datos reales de la época estival, en el cual se lleva a cabo un cultivo hidropónico de rosas. El problema es bastante complejo, por el elevado número de parámetros desconocidos del modelo, por la presencia de ruido de medida y dinámica no modelada considerables y por la utilización simultánea de cuatro normas sobre el error de identificación producido sobre las dos salidas

del modelo, sin embargo, la metodología de IR propuesta ha permitido resolver el problema de IR satisfactoriamente. Parte de estos resultados se han publicado en [79, 113].

9.2. Líneas futuras

Las líneas futuras de investigación pueden ir encaminadas a cubrir los siguientes aspectos:

- **Mejoras en los algoritmos ϵ-GA y $\not\epsilon$MOGA**. Incorporación de espacios de búsqueda variables, desarrollo de nuevos operadores genéticos, utilización de otras estructuras paralelas, hibridar los algoritmos con optimizadores locales para mejorar la precisión, etc.
- **Mejoras en la metodología propuesta**. Incorporar un capa de decisión (*decision maker*) que a partir del frente de Pareto permita determinar un modelo nominal de proyección interpolada atendiendo a aspectos de carácter cualitativo. Mejorar el procedimiento de determinación de las cotas asociadas a las normas del EI para que su elección resulte menos subjetiva.
- **IR de otros procesos**. En el caso del modelo de bloqueo de fármacos, es posible utilizar modelos más completos que incorporen la dinámica del propio potencial de acción. De esta manera, sería posible establecer la interacción entre el fármaco y el potencial de acción pudiendo analizar sus efectos directos.
- **Otras aplicaciones**. Utilización de los algoritmos ϵ-GA y $\not\epsilon$MOGA en la reducción de modelos. Por ejemplo, dado un modelo no lineal, determinar el FPS asociado a los parámetros de otro modelo (lineal o no lineal más sencillo) que reproduzca el comportamiento del modelo original con error acotado.
- **Control robusto no lineal**. Utilizar el FPS^* generado en la fase de IR para ajustar los parámetros de un controlador robusto (por ejemplo un PID) o abordar directamente la determinación de las acciones de control (por ejemplo de un controlador predictivo). Esto conllevaría, en ambos casos, la resolución de un problema min-max. Dado que el FPS^* es un conjunto finito, la resolución no sería excesivamente complicada aunque sí costosa desde el punto de vista computacional. Evidentemente, siempre sería posible mayorar el FPS^* obtenido mediante un hipervolumen (esfera, cubo, etc.) y utilizar los algoritmos clásicos, aunque esto supondría adoptar una postura más conservativa.

Apéndice A

Problemas de Optimización

A.1. Problemas multiobjetivo

En esta sección se muestran una serie de problemas de optimización [36] diseñados para el análisis de los algoritmos de optimización multiobjetivo. Dichos problemas se caracterizan, expresamente, por contener funciones a optimizar que son no lineales y plantear frentes de Pareto no convexos, discontinuos y no uniformes así como conjuntos de óptimos discontinuos, óptimos aislados, etc.

Para obtener la solución de los diferentes problemas, que a continuación se mostrarán, se ha aplicado búsqueda exhaustiva para explorar el espacio de búsqueda (para algunas de los problemas se han explorado más de un millón de puntos)[1].

A.1.1. Problema MOP1

El problema MOP1 es uno de los problemas típicos de evaluación de MOEAs, principalmente por cuestiones históricas, más que por la dificultad que entraña. Entre las características que incorpora este problema están:

- Es posible determinar analíticamente la solución y por lo tanto es muy fácil encontrar Θ_P.
- El frente de Pareto es convexo.
- Θ_P es una línea.

Las funciones a optimizar son $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)]$, donde:

$$\begin{aligned} J_1(\theta) &= \theta^2, \\ J_2(\theta) &= (\theta - 2)^2, \end{aligned}$$

$$-10^5 \leq \theta \leq 10^5.$$

La figura A.1 muestra el conjunto de óptimos de Pareto Θ_P^{MOP1} y el frente de Pareto correspondiente.

A.1.2. Problema MOP2

El problema MOP2 presenta las siguientes características:

- Es posible añadir arbitrariamente más variables de decisión sin alterar el frente de Pareto.

[1] Hay que tener en cuenta que este planteamiento sería inabordable desde un punto de vista de aplicación real.

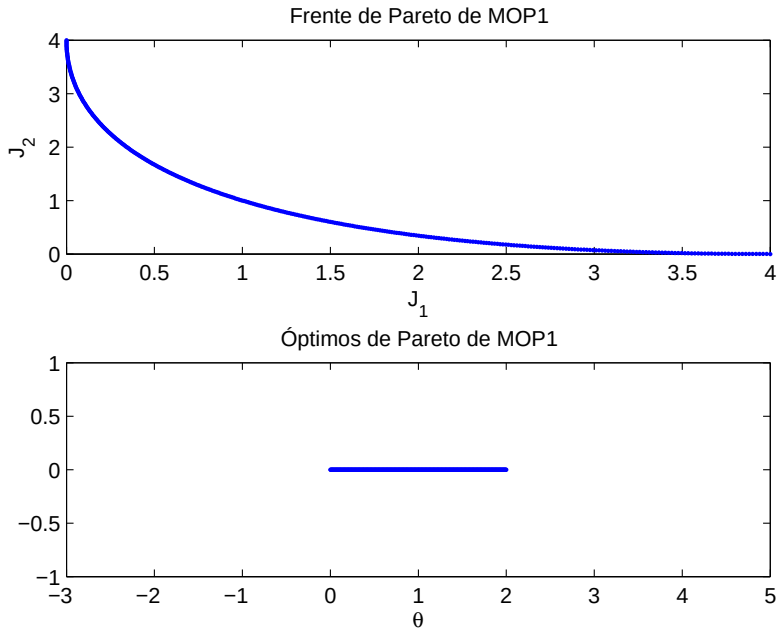

Figura A.1: Arriba frente de Pareto para el problema MOP1. Debajo el conjunto de óptimos Θ_P^{MOP1} que generada dicho frente.

- El frente de Pareto es concavo.
- Θ_P es una área (delgada y alargada) dentro del espacio de búsqueda.

Las funciones a optimizar son $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)]$, $\theta = [\theta_1, \theta_2, \theta_3]$ donde:

$$\begin{aligned} J_1(\theta) &= 1 - e^{-\sum_{i=1}^{n}\left(\theta_i - \frac{1}{\sqrt{n}}\right)^2}, \\ J_2(\theta) &= 1 - e^{-\sum_{i=2}^{n}\left(\theta_i + \frac{1}{\sqrt{n}}\right)^2}, \end{aligned}$$

$$-4 \leq \theta_i \leq 4; \quad n = 3.$$

La figura A.2 muestra el conjunto de óptimos de Pareto Θ_P^{MOP2} y el frente de Pareto correspondiente.

A.1.3. Problema MOP3

El problema MOP3 presenta las siguientes características:

- El frente de Pareto es discontinuo.

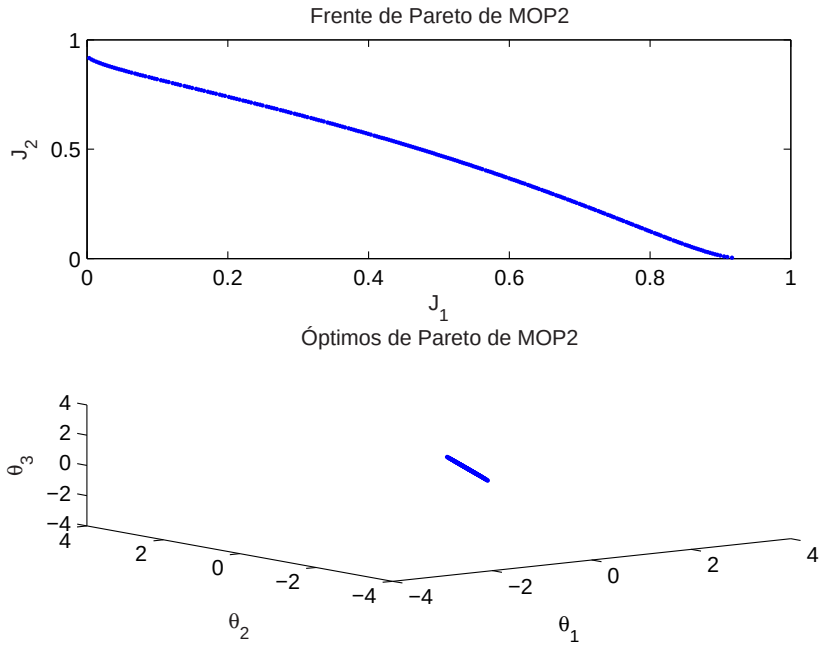

Figura A.2: Arriba frente de Pareto para el problema MOP2. Debajo el conjunto de óptimos Θ_P^{MOP2} que generada dicho frente.

- Θ_P es discontinuo también.

Las funciones a optimizar son $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)]$, $\theta = [\theta_1, \theta_2]$ donde:

$$\begin{aligned} J_1(\theta) &= \left(1 + (A_1 - B_1)^2 + (A_2 - B_2)^2\right), \\ J_2(\theta) &= (\theta_1 + 3)^3 + (\theta_2 + 1)^2, \end{aligned}$$

$$\begin{aligned} A_1 &= 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2), \\ A_2 &= 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2), \\ B_1 &= 0.5\sin(\theta_1) - 2\cos(\theta_1) + \sin(\theta_2) - 1.5\cos(\theta_2), \\ B_2 &= 1.5\sin(\theta_1) - \cos(\theta_1) + 2\sin(\theta_2) - 0.5\cos(\theta_2), \\ &-\pi \leq \theta_1 \leq \pi; \quad -\pi \leq \theta_2 \leq \pi. \end{aligned}$$

La figura A.3 muestra el conjunto de óptimos de Pareto Θ_P^{MOP3} y el frente de Pareto correspondiente.

A.1.4. Problema MOP4

El problema MOP4 presenta las siguientes características:

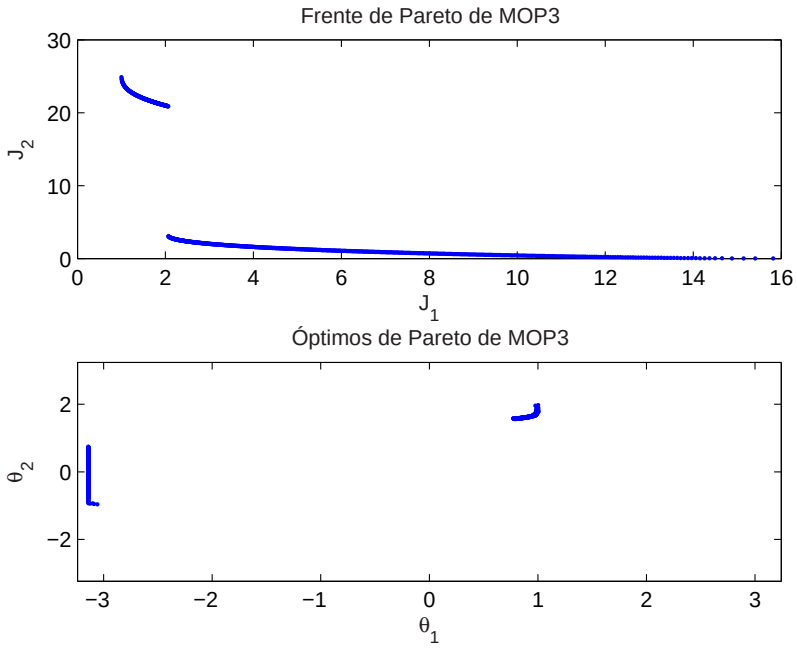

Figura A.3: Arriba frente de Pareto para el problema MOP3. Debajo el conjunto de óptimos Θ_P^{MOP3} que generada dicho frente.

- Es posible añadir arbitrariamente más variables de decisión, lo cual modifica ligeramente el frente de Pareto.
- El frente de Pareto es discontinuo.
- Θ_P presenta discontinuidad y asimetría.

Las funciones a optimizar son $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta)]$, $\theta = [\theta_1, \theta_2, \theta_3]$ donde:

$$\begin{aligned} J_1(\theta) &= \sum_{i=1}^{n-1} -10e^{\left(-0.2\sqrt{\theta_i^2+\theta_{i+1}^2}\right)}, \\ J_2(\theta) &= \sum_{i=1}^{n} \left(|\theta_i|^b + 5\sin(\theta_i)^b\right), \end{aligned}$$

$$-4 \leq \theta_i \leq 4; \quad n = 3; \quad a = 0.8; \quad b = 3.$$

La figura A.4 muestra el conjunto de óptimos de Pareto Θ_P^{MOP4} y el frente de Pareto correspondiente.

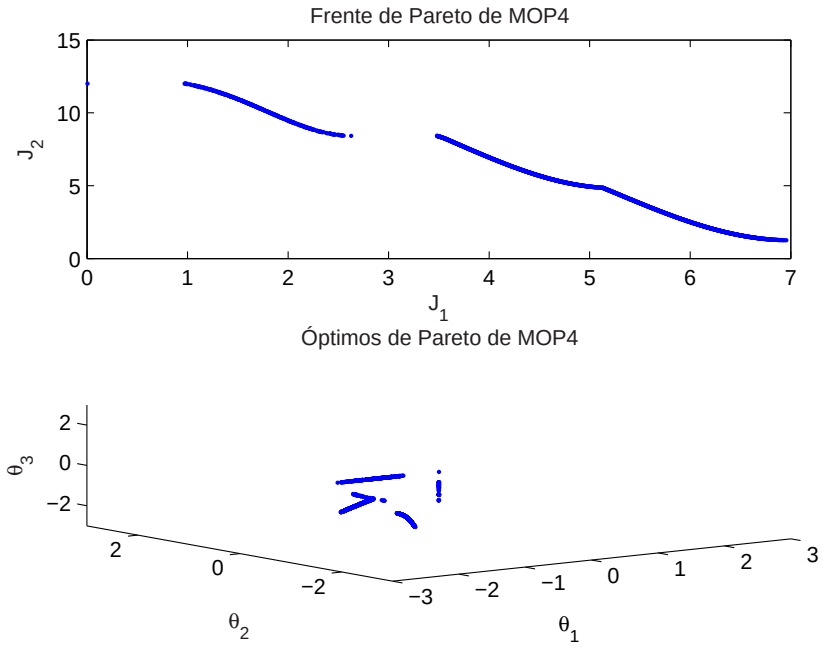

Figura A.4: Arriba frente de Pareto para el problema MOP4. Debajo el conjunto de óptimos Θ_P^{MOP4} que generada dicho frente.

A.1.5. Problema MOP5

El problema MOP5 presenta las siguientes características:

- Frente de Pareto continuo en el espacio tridimensional que se enrolla.
- Θ_P es discontinuo.

Las funciones a optimizar son $\mathbf{J}(\theta) = [J_1(\theta), J_2(\theta), J_3(\theta)]$, $\theta = [\theta_1, \theta_2]$ donde:

$$\begin{aligned} J_1(\theta) &= 0.5(\theta_1^2 + \theta_2^2) + \sin(\theta_1^2 + \theta_2^2), \\ J_2(\theta) &= \frac{(3\theta_1 - 2\theta_2 + 4)^2}{8} + \frac{(\theta_1 - \theta_2 + 1)^2}{27}, \\ J_3(\theta) &= \frac{1}{(\theta_1^2 + \theta_2^2 + 1)} - 1.1e^{(-\theta_1^2 - \theta_2^2)} + 0.2, \end{aligned}$$

$$-3 \leq \theta_1 \leq 3; \quad -3 \leq \theta_2 \leq 3.$$

La figura A.5 muestra el conjunto de óptimos de Pareto Θ_P^{MOP5} y el frente de Pareto correspondiente.

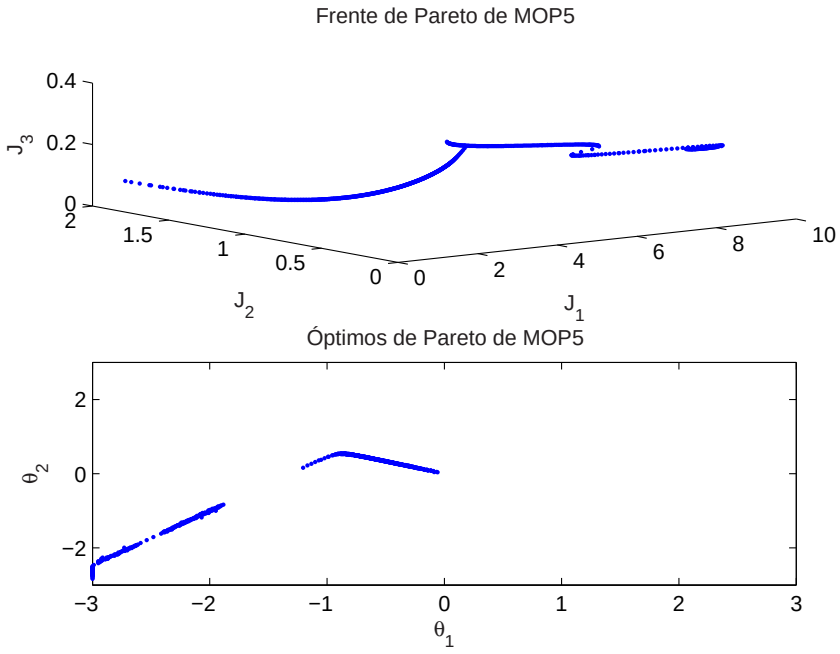

Figura A.5: Arriba frente de Pareto para la función de test MOP5. Debajo el conjunto de óptimos Θ_P^{MOP5} que generada dicho frente.

A.2. Funciones multimodales

En esta sección se muestran una serie de problemas de optimización multimodal [20]. Dichos problemas se caracterizan expresamente por contener funciones a optimizar que son no lineales y plantear infinitos óptimos globales y además óptimos locales.

Para obtener la solución de los diferentes problemas, que a continuación se mostrarán, se ha aplicado búsqueda exhaustiva. Para explorar el espacio de búsqueda (para algunas de los problemas se han explorado más de un millón de puntos) utilizando para ello un coste computacional muy elevado.

A.2.1. Problema OP1

El problema OP1 es un problema unidimensional propuesto por el autor de la tesis. Entre las características que incorpora este problema están:

- El conjunto de mínimos globales es inconexo.
- El mínimo global se encuentra en 0.
- Presenta mínimos locales cercanos a los globales.
- Gran parte del espacio de búsqueda es una llanura.

La función a optimizar es la siguiente:

$$J(\theta) = \begin{cases} f(\theta), & f(\theta) > 0 \\ 0, & f(\theta) <= 0 \end{cases}, \tag{A.1}$$

$$f(\theta) = \cos(5\pi \cdot |\theta|) \cdot e^{(-2\cdot|\theta|)} + 0.2,$$

$$-50 \leq \theta \leq 50.$$

La figura A.6 muestra la función a optimizar y el conjunto de óptimos globales Θ^*_{OP1}.

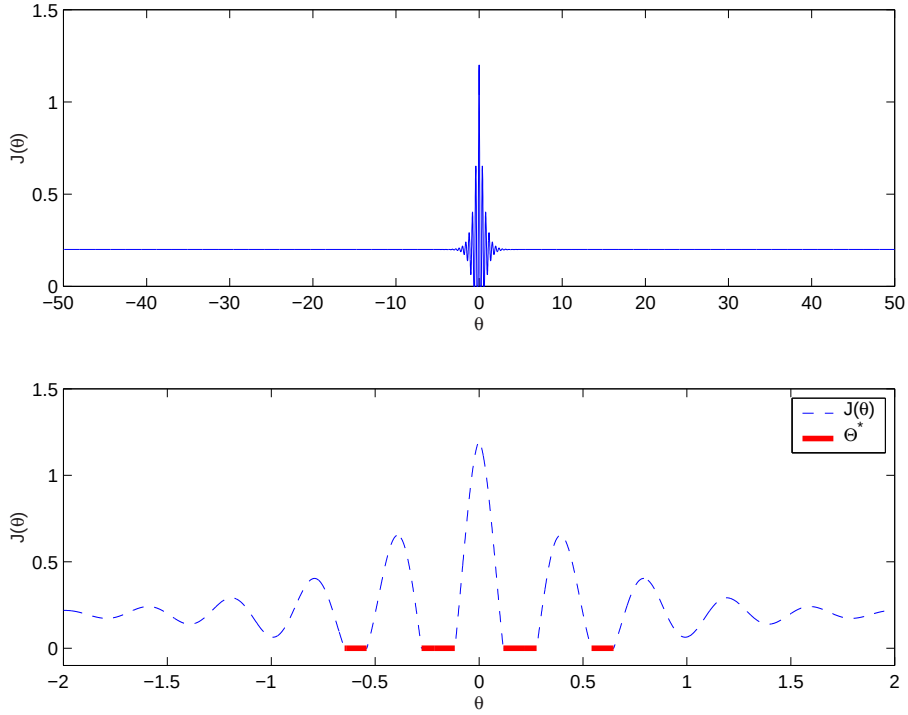

Figura A.6: Arriba la función del problema OP1. Debajo la función y el conjunto de mínimos globales Θ^*_{OP1}.

A.2.2. Problema OP2

El problema OP2 es un problema bidimensional derivado de la función Rastrigin. Entre las características que incorpora este problema están:

- El conjunto de mínimos globales es inconexo.
- El mínimo global se encuentra en 0.
- Presenta mínimos locales cerca y lejos de los globales.

La función a optimizar es la siguiente:

$$J(\theta) = |17 + \theta_1^2 - 10 \cdot \cos(\theta_1) + \theta_2^2 - 10 \cdot \cos(2\pi \cdot \theta_2)|, \tag{A.2}$$

$$-20 \leq \theta_1 \leq 20, -20 \leq \theta_2 \leq 20.$$

La figura A.7 muestra el conjunto de óptimos globales Θ^*_{OP2}.

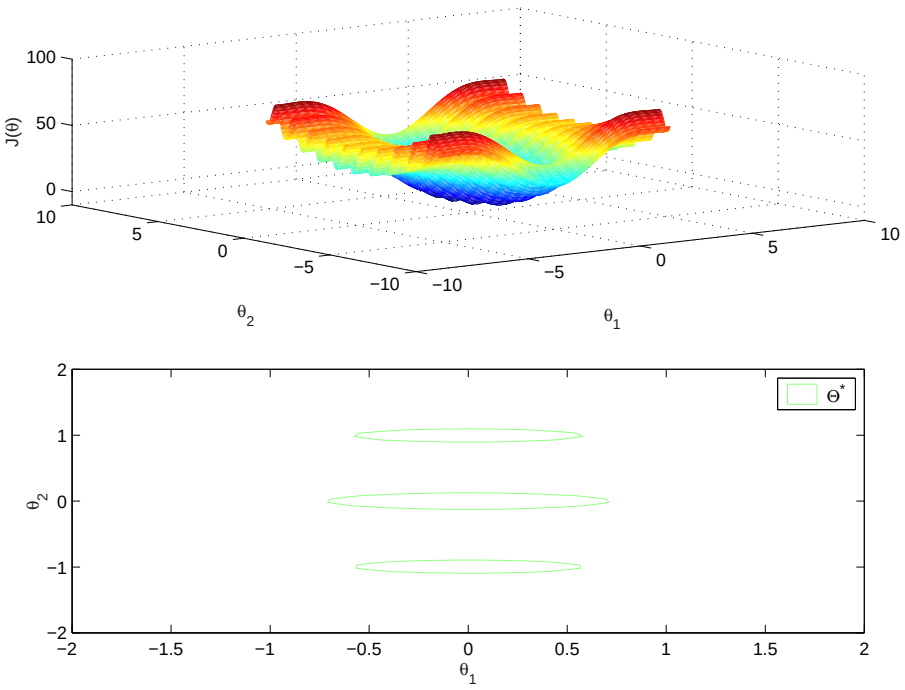

Figura A.7: Arriba la función del problema OP2. Debajo el conjunto de mínimos globales Θ^*_{OP2}.

A.2.3. Problema OP3

El problema OP3 es un problema bidimensional derivado de la función Shekels Foxholes. Entre las características que incorpora este problema están:

- El conjunto de mínimos globales es conexo.
- El mínimo global se encuentra en 1.
- Presenta mínimos locales cerca y lejos de los globales.
- Presenta muchas zonas con llanuras.

La función a optimizar es la siguiente:

$$J(\theta) = e^{\frac{|f(\theta)-125.0}{100}|}, \tag{A.3}$$

$$\frac{1}{f(\theta)} = \frac{1}{500} + \sum_{j=1}^{25} \frac{1}{c_j + \sum_{i=1}^{2}(\theta_i - a_{ij})^6},$$

$$(a_{ij}) = \begin{pmatrix} b_1 & b_1 & b_1 & b_1 & b_1 \\ -32 \cdot u & -16 \cdot u & 0 \cdot u & 16 \cdot u & 32 \cdot u \end{pmatrix},$$

$$(b_1) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 \end{pmatrix},$$

$$(u) = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix},$$

$$c_j = 20 + a_{1j}^2 - 10\cos(2\pi \cdot a_{1j}) + a_{2j}^2 - 10\cos(2\pi \cdot a_{2j}),$$

$$-65.536 \leq \theta 1 \leq 65.536, -65.536 \leq \theta_2 \leq 65.536.$$

La figura A.8 muestra el conjunto de óptimos globales Θ^*_{OP3}.

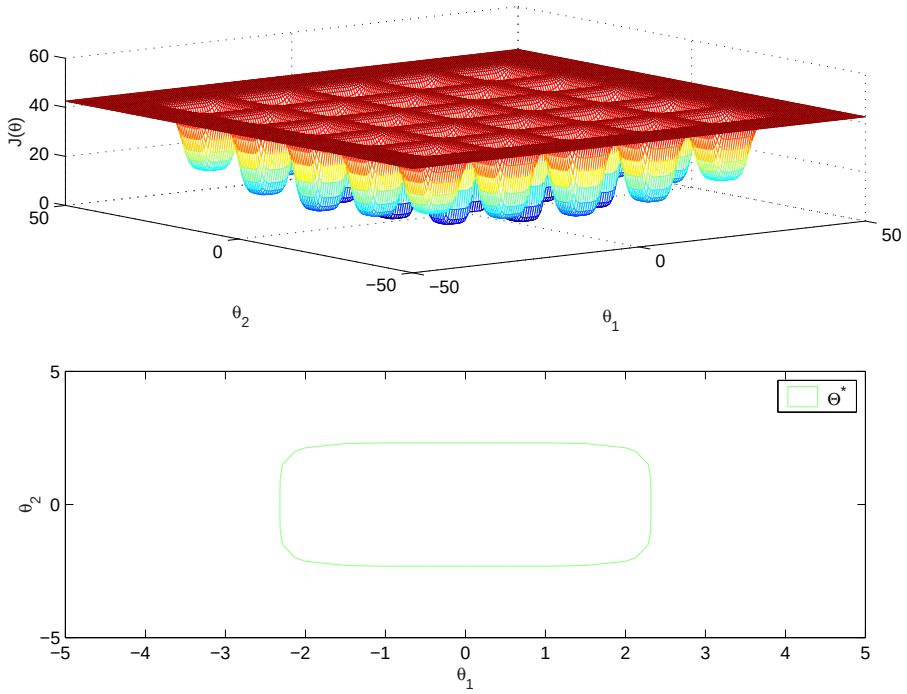

Figura A.8: Arriba la función del problema OP3. Debajo el conjunto de óptimos globales Θ^*_{OP3}.

A.2.4. Problema OP4

El problema OP4 es un problema bidimensional derivado de la función Schwefel. Entre las características que incorpora este problema están:

- El conjunto de mínimos globales es conexo y se encuentra en un rincón del espacio de búsqueda.
- El mínimo global se encuentra en 1.
- Presenta mínimos locales cerca y lejos de los globales.

La función a optimizar es la siguiente:

$$J(\theta) = e^{\frac{|f(\theta)+775.0|}{1000}}, \tag{A.4}$$

$$f(\theta) = -\theta_1 \cdot \sin(|\theta_1|)^{0.5} - \theta_2 \cdot \sin(|\theta_2|)^{0.5},$$

$$-250.0 \leq \theta 1 \leq 500.0, -250.0 \leq \theta_2 \leq 500.0.$$

La figura A.9 muestra el conjunto de óptimos globales Θ^*_{OP4}.

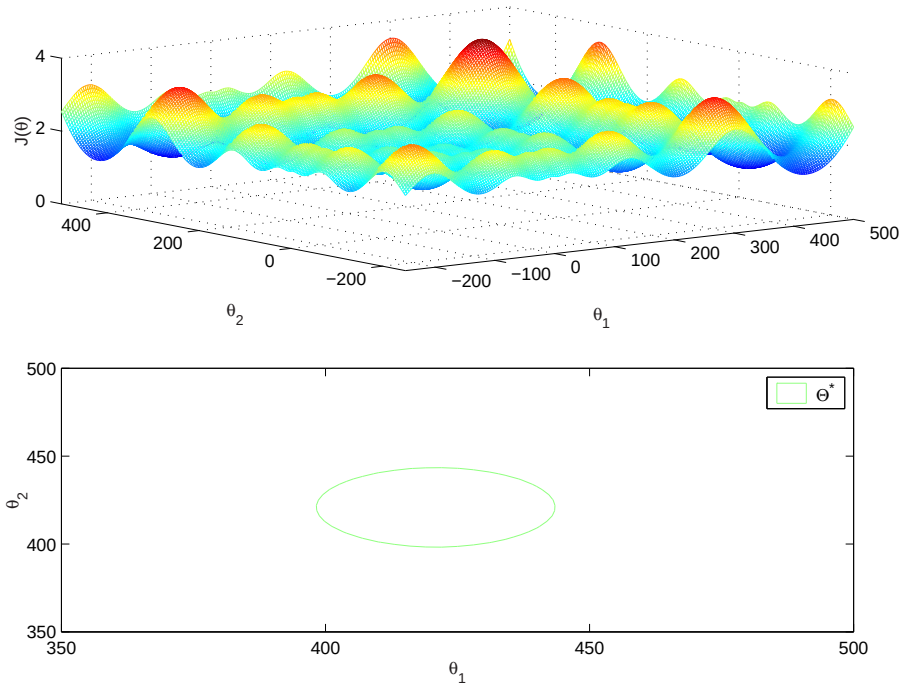

Figura A.9: Arriba la función del problema OP4. Debajo el conjunto de óptimos globales Θ^*_{OP4}.

A.2.5. Problema OP5

El problema OP5 es un problema tridimensional derivado de la función Schwefel. Entre las características que incorpora este problema están:

- El conjunto de mínimos globales es conexo y se encuentra en un rincón del espacio de búsqueda.
- El mínimo global se encuentra en 1.
- Presenta mínimos locales cerca y lejos de los globales.

La función a optimizar es la siguiente:

$$J(\theta) = e^{\frac{|f(\theta)+1150.0|}{1000}}, \tag{A.5}$$

$$f(\theta) = -\theta_1 \cdot \sin(|\theta_1|)^{0.5} - \theta_2 \cdot \sin(|\theta_2|)^{0.5} - \theta_3 \cdot \sin(|\theta_3|)^{0.5},$$

$$-250.0 \leq \theta 1 \leq 500.0, -250.0 \leq \theta_2 \leq 500.0, -250.0 \leq \theta_3 \leq 500.0.$$

La figura A.10 muestra el conjunto de óptimos globales Θ^*_{OP5}.

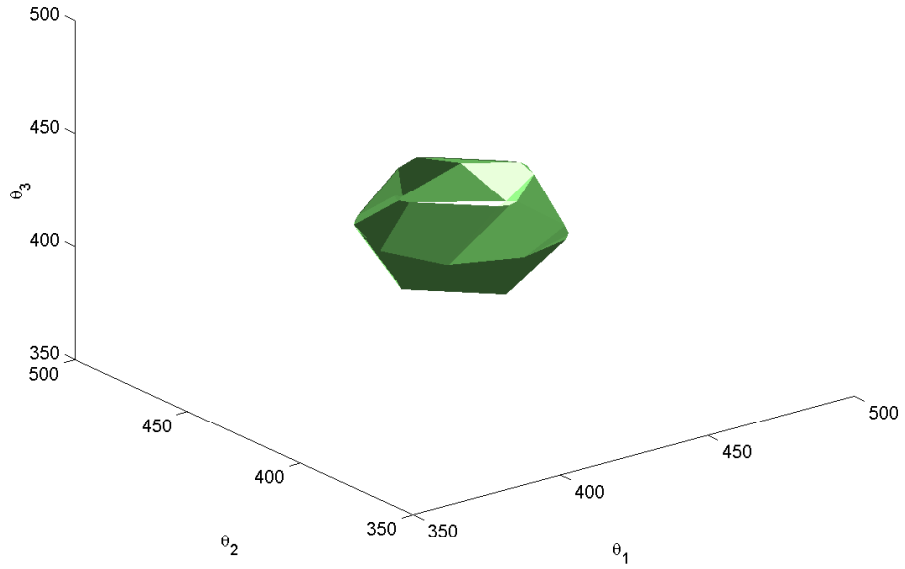

Figura A.10: Conjunto de óptimos globales Θ^*_{OP5}.

Apéndice B

Modelo del Invernadero y Datos para la Identificación

A continuación, se presenta un apartado dedicado a notación, donde se enumeran alfabéticamente las variables y parámetros presentes en las ecuaciones de este Apéndice. En particular, se presentan los rangos de los parámetros inciertos y que han sido objeto de identificación en esta tesis. Seguidamente se muestran las ecuaciones que complementan el modelo presentado en la sección 8.2. Para terminar, se presentan los datos que se utilizarán el proceso de identificación y validación del modelo climático.

B.1. Notación

Para los parámetros inciertos se da el rango de variación.

- a: Constante para el flujo de renovación, [0.0005, 0.01].
- A: Área de viento, 130 m^2.
- A_c: Coeficiente de pérdidas, [2, 20].
- A_i: Superficie del invernadero, 240 m^2.
- C_m: Capacidad calorífica de la masa térmica, [1e5, 3e5] $J^oC^{-1}m^{-2}$.
- c_p: Calor esp. del aire, 1003 $J\ Kg^{-1}\ ^oC^{-1}$.
- C_{sat}: Coeficiente de saturación del aire, adimensional.
- D_i: Deficit de saturación, KPa.
- E: Evapotranspiración cultivo, $Kg_{H_2O}\ s^{-1}$.
- fog: Nebulización, $Kg_{H_2O}\ s^{-1}$.
- fog_{max}: Nebulización máxima, [0.001, 0.02] $Kg_{H_2O}\ s^{-1}$.
- F_v: Flujo de renovación del vapor de agua en el aire, $Kg_{H_2O}\ s^{-1}$.
- gwb: Conductancia de la capa límite, [0.001, 0.05] $m\ s^{-1}$.
- gws: Conductancia estomática, $m\ s^{-1}$.
- gws_{max}: Conductancia estomática máxima, [0.01, 0.03] $m\ s^{-1}$.
- gws_{min}: Conductancia estomática mínima, [0.0001, 0.005] ms^{-1}.
- G: Caudal de renovación del aire, $m^3\ s^{-1}$.
- G_o: Constante de fugas, [0.0, 0.01].
- h_m: Coeficiente de conductividad entre masa térmica y aire, [1, 20] $W\ m^{-1}\ ^o\ K^{-1}$.

- $\hat{HR}_i$: Humedad relativa interior, %.
- HR_o: Humedad relativa exterior, %.
- H_i: Humedad absoluta int., $Kg_{H_2O}\ Kg_{aire}^{-1}$.
- H_o: Humedad absoluta ext., $Kg_{H_2O}\ Kg_{aire}^{-1}$.
- H_{sat}: Humedad absoluta de saturación, $Kg_{H_2O}\ Kg_{aire}^{-1}$.
- k: Coeficiente de extinción de la radiación, [0.1, 0.7].
- k_a: Coeficiente de conductividad entre masa térmica y suelo, [0.5, 10] $W\ m^{-1\ o}\ K^{-1}$.
- L: Índice de área foliar, [0.5, 2] $m_{hojas}^2\ m_{suelo}^{-2}$.
- MV_{fog}: Nebulización, %.
- MV_W: Calefacción, %.
- MV_α: Apertura de la ventana, %.
- $psat$: Presión de saturación del vapor de agua, KPa.
- P: Presión atmosférica, 98.1 KPa.
- Q_{cc}: Pérdidas de energía por conducción y convección, W.
- Q_e: Pérdidas de energía debidas a la evapotranspiración del cultivo, W.
- Q_f: Pérdidas hacia el fondo del suelo, W.
- Q_m: Intercambio de energía con la masa térmica, W.
- Q_n: Pérdidas de energía por nebulización, W.
- Q_s: Energía solar suministrada al aire, W.
- Q_{sm}: Energía almacenada por la masa térmica durante el día, W.
- Q_v: Intercambio de energía debido a la ventilación, W.
- Rn: Radiación solar absorbida por las plantas, $W\ m^{-2}$.
- S_o: Radiación solar, $W\ m^{-2}$.
- $\hat{T}_i$: Temperatura interior, ^oC.
- T_m: Temperatura masa térmica, ^oC.
- T_o: Temperatura exterior, ^oC.
- T_{ref}: Temperatura del suelo a la profundidad de referencia, [10, 20] ^oC.

- V: Velocidad del viento, $m\ s^{-1}$.
- v_i: Volumen del invernadero, 850 m^3.
- W: Potencia de la calefacción, W.
- W_{max}: Potencia máxima de la calefacción, 5000 W.
- z_{ref}: Profundidad de referencia, 6 m.
- α: Ángulo de la ventana, ^{o}.
- α_m: Factor de calor absorbido por la masa térmica, [0.01, 0.3].
- α_{max}: Ángulo máximo de la ventana, 12 ^{o}.
- Δ: Pendiente de saturación del vapor de agua KPa^oC^{-1}.
- γ: Constante Psicométrica, 0.066 KPa^oC^{-1}.
- λ: Calor latente de vaporización, $J\ Kg^{-1}$.
- ρ: Densidad del aire, 1.25 $Kg_{aire}\ m^{-3}$.
- τ: Coeficiente de transmisión del invernadero, [0.3, 0.9].

B.2. Ecuaciones complementarias

Control de apertura de la ventana:

$$\alpha = \frac{MV_\alpha}{100}\alpha_{max}. \tag{B.1}$$

Control de la calefacción:

$$W = \frac{MV_W}{100}W_{max}. \tag{B.2}$$

Control de la nebulización:

$$fog = \frac{MV_{fog}}{100}fog_{max}. \tag{B.3}$$

Flujo de renovación del vapor de agua en el aire:

$$F_v = \rho G(H_o - H_i). \tag{B.4}$$

Caudal de renovación del aire [23]:

$$G = AV(a\alpha + G_o). \tag{B.5}$$

Coeficiente de saturación del aire:

$$C_{sat} = \begin{cases} 1 & H_i < H_{sat} \\ 0 & H_i = H_{sat} \end{cases}. \tag{B.6}$$

Relación entre humedad absoluta y relativa[1]:

$$HR = \begin{cases} 100 & HR > 100 \\ HR & HR \leq 100 \end{cases},$$

$$HR = \frac{100H \cdot P}{0.611psat(T)}, \tag{B.7}$$

$$p_{sat}(T) = 0.61\left[1 + 1.414\sin(5.82e^{-3}T)\right]^{8.827}. \tag{B.8}$$

Evapotranspiración del cultivo [128]:

$$E = \frac{A_i(\Delta Rn + 2L\rho\, c_p\, D_i\, gwb)}{\left[\Delta + \gamma\left(1 + \frac{gwb}{gws}\right)\right]\lambda}, \tag{B.9}$$

$$\Delta = p_{sat}(\hat{T}_i + 0.5) - p_{sat}(\hat{T}_i - 0.5), \tag{B.10}$$

$$Rn = (1 - e^{kL})\tau So, \tag{B.11}$$

$$D_i = p_{sat}(\hat{T}_i)\left[1 - \frac{\hat{H}R_i}{100}\right], \tag{B.12}$$

$$\lambda = (3.1468 - 0.002365(\hat{T}_i + 273)10^6, \tag{B.13}$$

$$\begin{aligned} gws &= gws_{min} + (gws_{max} - gws_{min}) \cdot \\ &\quad \cdot \left[1 - exp\left(-\frac{\tau So}{160}\right)\right] g_D, \end{aligned} \tag{B.14}$$

$$g_D = \begin{cases} \frac{0.39}{0.029 + D_i} & D_i \geq 0.361 \\ 1 & D_i < 0.361 \end{cases}.$$

[1]Según los casos, corresponde al interior $(\hat{T}_i, \hat{H}R_i, H_i)$ o exterior T_o, HR_o, H_o del invernadero. Permite también el cálculo de la humedad absoluta de saturación H_{sat} correspondiente a $HR = 100\,\%$.

Energía solar suministra al aire:

$$Q_s = A_i \tau So. \tag{B.15}$$

Intercambio de energía por conducción y convección:

$$Q_{cc} = A_i A_c (\hat{T}_i - T_o). \tag{B.16}$$

Pérdidas debidas a la evapotranspiración:

$$Q_e = \lambda E. \tag{B.17}$$

Intercambio de energía debido a la ventilación:

$$Q_v = \rho c_p G (\hat{T}_i - T_o). \tag{B.18}$$

Pérdidas por nebulización:

$$Q_n = \lambda fog. \tag{B.19}$$

Intercambio de energía entre la masa térmica y el aire interior:

$$Q_m = A_i h_m (T_m - \hat{T}_i). \tag{B.20}$$

Energía almacenada por la masa térmica durante el día:

$$Q_{sm} = \alpha_m Q_s. \tag{B.21}$$

Pérdidas de energía hacia el fondo del suelo:

$$Q_f = A_i k_a \left(\frac{T_m - T_{ref}}{z_{ref}} \right). \tag{B.22}$$

B.3. Datos para la identificación y validación

La figura B.1 muestra la evolución temporal de la temperatura y la humedad relativa interior del invernadero para los días 11 y 15 de junio de 2002, mientras que en la figura B.2 se muestra la evolución de la señales de entrada para dichos días. Estos son los datos $\Omega_{ide} = \{Y_{ide}(t), U_{ide}(t)\}$ que serán utilizados en el proceso de identificación.

Las figuras B.3 y B.4 muestran los datos Ω_{val1} correspondientes al día 20 de junio de 2002, las figuras B.5 y B.6 los datos Ω_{val2} correspondientes al día 28 de julio de 2002, las figuras B.7 y B.8 los datos Ω_{val3} correspondientes al día 22 de agosto de 2002 y las figuras B.9 y B.10 los datos Ω_{val4} correspondientes al día 8 de septiembre de 2002.

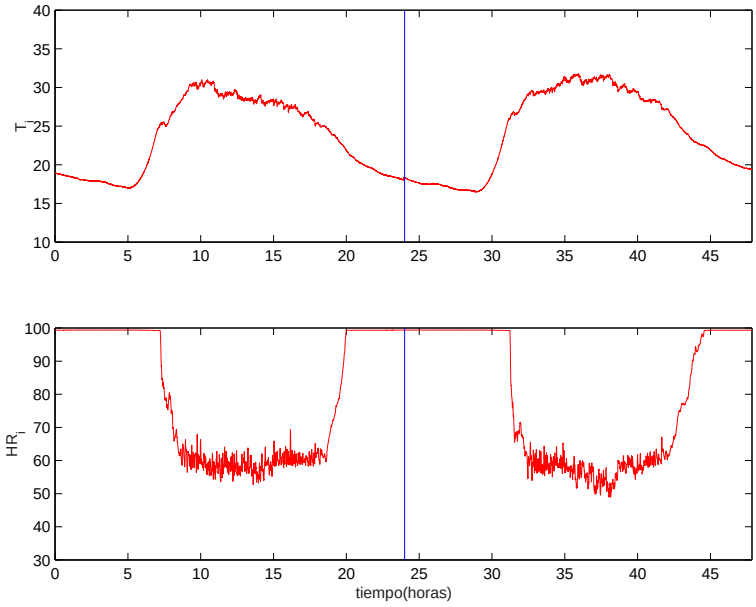

Figura B.1: Datos $Y_{ide}(t)$ para la identificación. Arriba la temperatura interior (^oC) y debajo la humedad relativa interior ($\%$) para los días 11 y 15 de junio de 2002. $N = 11520$ (longitud del experimento) con $T = 15seg.$ (periodo de muestreo).

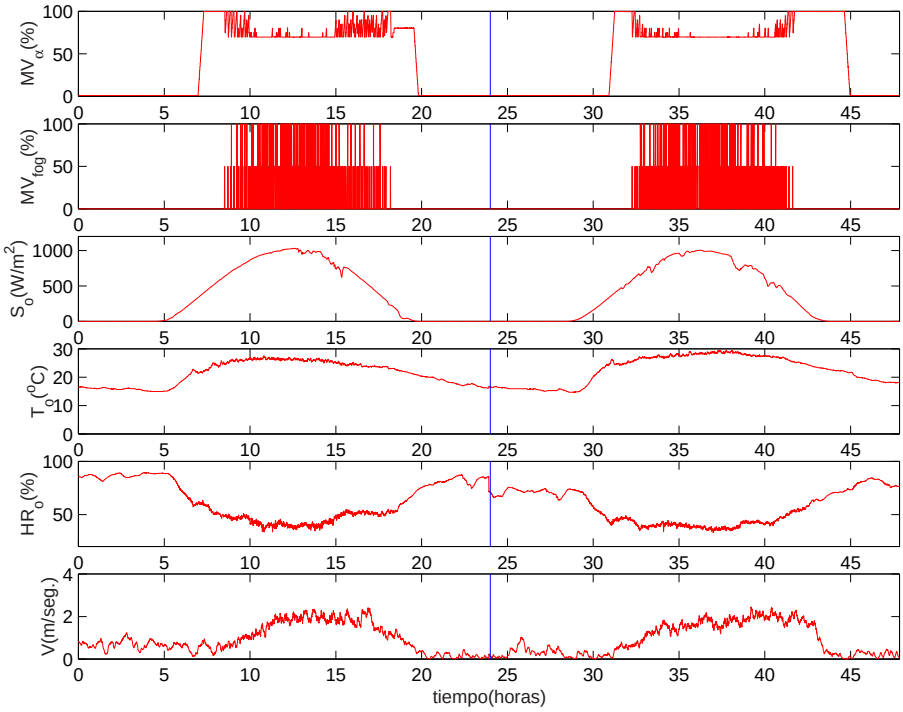

Figura B.2: Datos $U_{ide}(t)$ para la identificación. Desde arriba hacia abajo: Apertura de la ventana ($\%$), nebulización ($\%$), radiación solar (W/m^2), temperatura exterior (^oC), humedad relativa exterior ($\%$) y velocidad del viento ($m/seg.$) para los días 11 y 15 de junio de 2002. $N = 11520$ y $T = 15seg.$

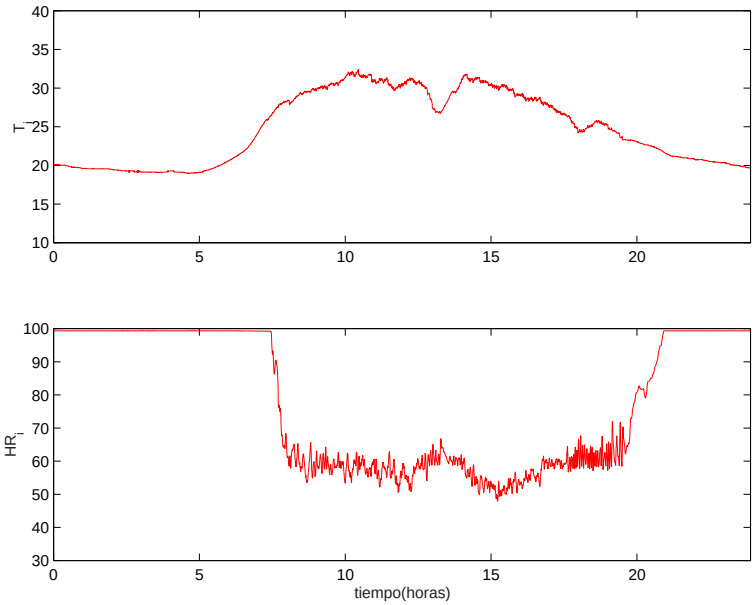

Figura B.3: Datos $Y_{val1}(t)$ para la validación. Arriba la temperatura interior (^oC) y debajo la humedad relativa interior ($\%$) del día 20 de junio de 2002. $N = 5760$ con $T = 15seg.$

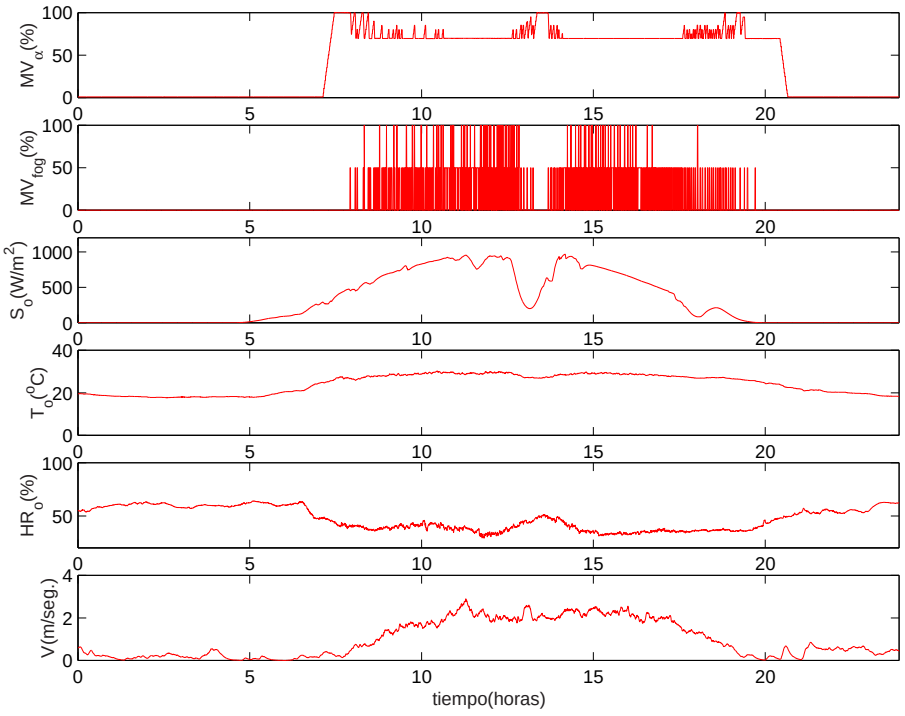

Figura B.4: Datos $U_{val1}(t)$ para la validación. Desde arriba hacia abajo: Apertura de la ventana ($\%$), nebulización ($\%$), radiación solar (W/m^2), temperatura exterior (^oC), humedad relativa exterior ($\%$) y velocidad del viento ($m/seg.$) del día 20 de junio de 2002. $N = 11520$ y $T = 15seg.$

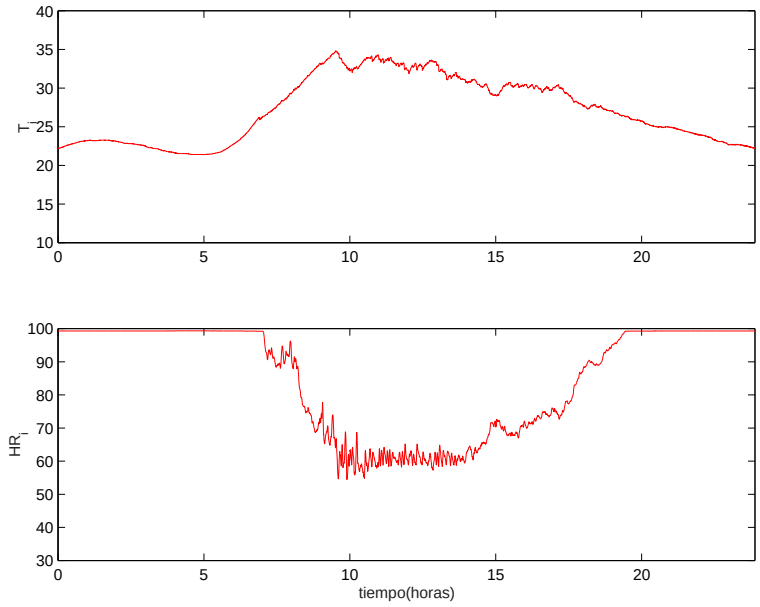

Figura B.5: Datos $Y_{val2}(t)$ para la validación. Arriba la temperatura interior (^oC) y debajo la humedad relativa interior ($\%$) del día 28 de julio de 2002. $N = 5760$ con $T = 15seg.$

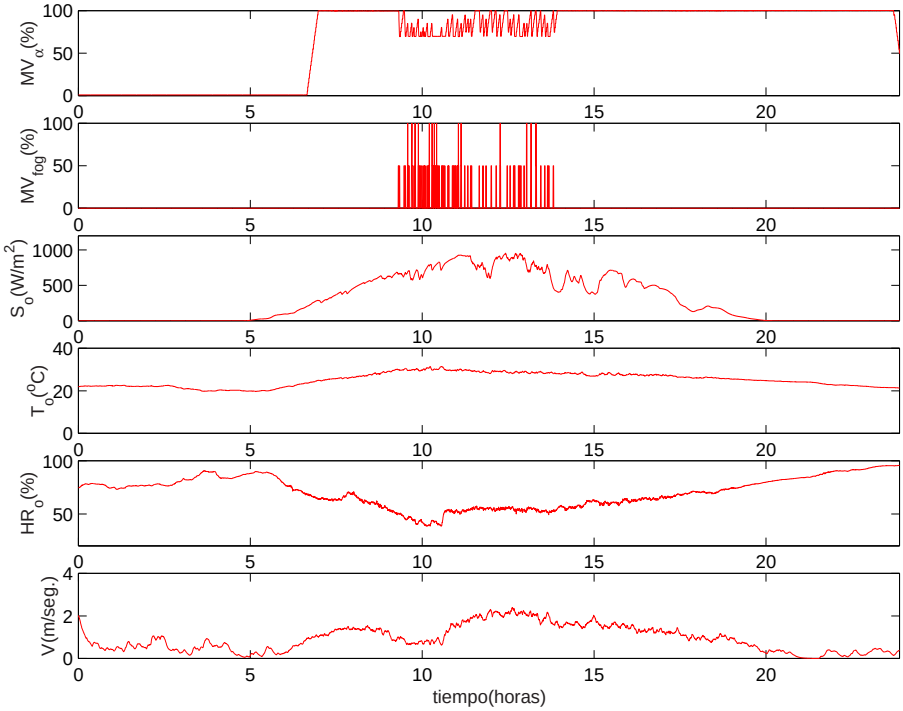

Figura B.6: Datos $U_{val2}(t)$ para la validación. Desde arriba hacia abajo: Apertura de la ventana ($\%$), nebulización ($\%$), radiación solar (W/m^2), temperatura exterior (^oC), humedad relativa exterior ($\%$) y velocidad del viento ($m/seg.$) del día 28 de julio de 2002. $N = 11520$ y $T = 15seg.$

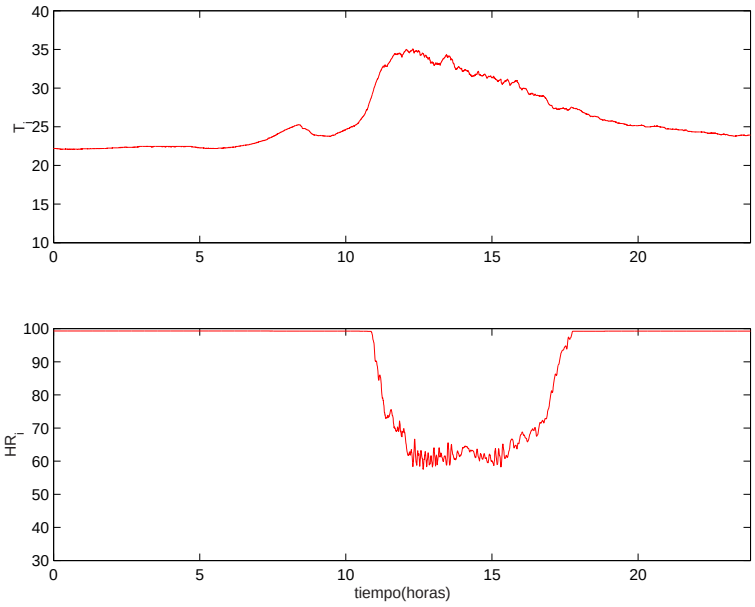

Figura B.7: Datos $Y_{val3}(t)$ para la validación. Arriba la temperatura interior (^oC) y debajo la humedad relativa interior ($\%$) del día 22 de agosto de 2002. $N = 5760$ con $T = 15seg.$

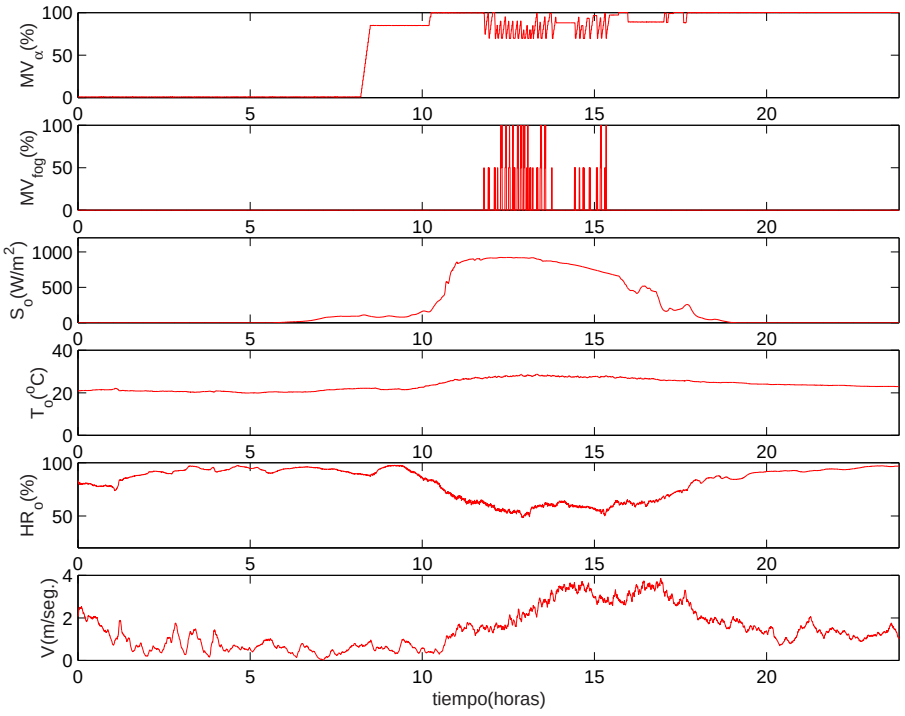

Figura B.8: Datos $U_{val3}(t)$ para la validación. Desde arriba hacia abajo: Apertura de la ventana ($\%$), nebulización ($\%$), radiación solar (W/m^2), temperatura exterior (^oC), humedad relativa exterior ($\%$) y velocidad del viento ($m/seg.$) del día 22 de agosto de 2002. $N = 11520$ y $T = 15seg.$

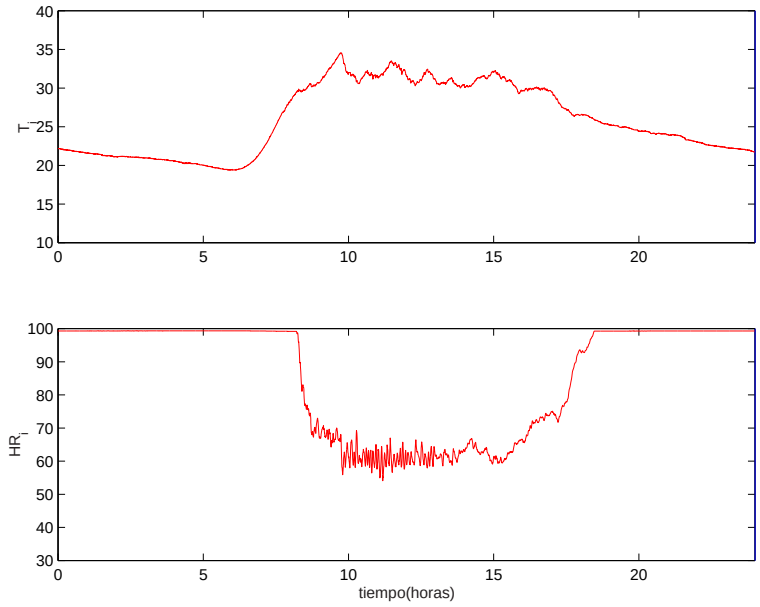

Figura B.9: Datos $Y_{val4}(t)$ para la validación. Arriba la temperatura interior (^oC) y debajo la humedad relativa interior ($\%$) del día 8 de septiembre de 2002. $N = 5760$ con $T = 15seg.$

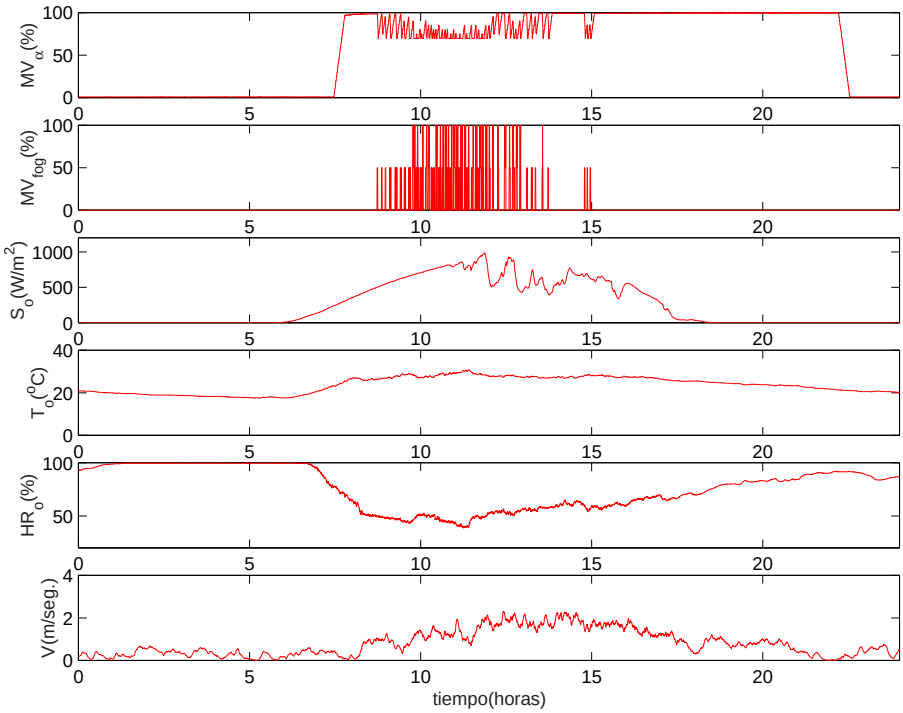

Figura B.10: Datos $U_{val4}(t)$ para la validación. Desde arriba hacia abajo: Apertura de la ventana ($\%$), nebulización ($\%$), radiación solar (W/m^2), temperatura exterior (^oC), humedad relativa exterior ($\%$) y velocidad del viento ($m/seg.$) del día 8 de septiembre de 2002. $N = 11520$ y $T = 15seg.$

Bibliografía

[1] H. Adeli and N. Cheng. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1):104–118, 1994.

[2] J. Alander. An indexed bibliography of Genetic Algorithms & Pareto and constrained optimization. Technical report, Dpt. of Information Technology, University of Vaasa, 2002.

[3] J.T. Alander. On optimal population size of genetic algorithms. *Proceedings CompEuro 92*, pages 65–70, 1992.

[4] D. Albright, I. Seginer, L.S. Marsh, and A. Oko. In situ thermal calibration of unventilated greenhouses. *Journal of Agricultural Engineering Research*, 31:265–281, 1985.

[5] O.B. Augusto, S. Rabeau, Ph. Dépincé, and F. Bennis. Multi-objective genetic algorithms: a way to improve the convergence rate. *Engineering applications of artificial intelligence*, En Prensa, 2006.

[6] T. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. *Foundations of Intelligent Systems, 9th International Symposium*, pages 158–167, 1996.

[7] T. Bäck and H.P. Schwefel. *Evolution strategies I: Variants and their computational implementation*, chapter 6, pages 111–126. Genetic Algorithms in Engineering and Computer Science. John wiley & sons ltd. edition, 1995.

[8] Thomas Bäck. *Evolutionary Algorithms in theory and Practice*. Oxford university Press, New York, 1996.

[9] E. Bai, K. Nagpal, and R. Tempo. Bounded-error parameter estimation: Noise models and recursive algorithms. *Automatica*, 32(7):985–999, 1996.

[10] E. Bai, Y. Ye, and R. Tempo. Bounded error parameter estimation: A survey. *IEEE Transaction on Automatic Control*, 44(6):1107–1117, 1999.

[11] M. Baille, A.Baille, and D. Delmon. Microclimate and transpiration of a greenhouse rose crop. *Agric. Forest Meteor.*, 71:83–87, 1994.

[12] P. Bak. *How nature works.* Copernicus, Springer-Verlag, 1996.

[13] J. E. Baker. Adaptative selection methods for genetic algorithms. In *Proc. First International Conference on Genetic Algorithms*, pages 101–111, 1985.

[14] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proc. Second International Conference on Genetic Algorithms*, pages 14–22, 1987.

[15] S.M. Batill. *Course: ME/AE 446. Finite Element Methods in Structural Analysis. Planar truss applications. www.nd.edu.* 1995.

[16] J. Bean. Genetics and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

[17] G. Belforte, B. Bona, and V. Cerone. Parameter estimation algorithms for a set-membership description of uncertainty. *Automatica*, 26(5):887–898, 1990.

[18] G. Belforte and M. Milanese. Uncertainty interval evaluation in presence of unknown-but-bounded errors. Nonlinear families of models. *Proc. of 1st IASTED Internat. Symp. Modelling, Identification, Control*, pages 75–79, 1981.

[19] F.X. Blasco. Controlador predictivo optimizado con algoritmos genéticos (GAGPC). Extensiones. Technical Report , Dept. Ingeniería de Sistemas Computadores y Automática. U. Politécnica de Valencia, 1995.

[20] F.X. Blasco. *Control predictivo basado en modelos mediante la incorporación de técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables*. PhD thesis, Universidad Politécnica de Valencia, Valencia, 1999.

[21] X. Blasco, J.M. Herrero, M. Martínez, and J. Senent. Nonlinear parametric model identification with genetic algorithms. Application to thermal process. *Lecture notes in computer science*, 2084, 2001.

[22] T. Boulard and A. Baille. A simple greenhouse climate control model incorporating effects of ventilation and evaporative cooling. *Agricultural and Forest Meteorology*, (65):145–157, 1993.

[23] T. Boulard and B. Draoui. Natural ventilation of greenhouse with continuos roof vents: Measurements and data analysis. *Journal of Agricultural Engineering Research*, (61):27–36, 1995.

[24] T. Boulard, B. Draouiand, and F. Neirac. Calibration and validation of a greenhouse climate control model. *Acta Horticulture*, 406:49–61, 1996.

[25] V. Broman and M.J. Shensa. A compact algorithm for the intersection and approximation of n-dimensional polytopes. *Math. and Computers in Simulation*, 32:469–480, 1990.

[26] E. Cantú-Paz. A summary of research on parallel genetic algorithms. Technical Report 95007, Illinois Genetic Algorithms Laboratory. IlliGAL, July 1995.

[27] K. E. Cardona. Modelización del comportamiento eléctrico del fármaco antiarrítmico clase ib- lidocaína en células del miocardio en cobaya y conejo. Technical report, Departamento de Ingeniería Electrónica, 2005.

[28] R.A. Caruana and J.D. Schaffer. Representation and hidden bias: Gray vs. binary coding. In *6th Int. Conf Machine Learning*, pages 153–161, 1988.

[29] L. Chisci, A. Garulli, A. Vicino, and G. Zappa. Block recursive parallelotopic bounding in set membership identification. *Automatica*, 34(1):15–22, 1998.

[30] C. Coello. *An empirical study of evolutionary techniques for multiobjective optimization in engineering design.* PhD thesis, Department of computer science, 1996.

[31] C. Coello. Introducción a la computación evolutiva. 2001.

[32] C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in applied Mechanics and Engineering*, 191:1245–1287, 2002.

[33] C. Coello. Recent trends in evolutionary multiobjective oprimization. *Evolutionary Multiobjective Optimization: Theoretical Advances And Applications*, pages 7–32, 2005.

[34] C. Coello, A. Christiansen, and H. Hernández. Multiobjective design optimization of counterweight balancing of a robot arm using genetic algorithm. *Proc. of the 7th International Conference on Tools with Artificial Intelligence*, pages 20–23, 1999.

[35] C. Coello, G. Toscano, and E.Mezura. Current and future research trends in evolutionary multiobjective optimization. *Information Processing with Evolutionary Algorithms: From Industrial Applications to Academic Speculations*, pages 213–231, 2005.

[36] C. Coello, D. Veldhuizen, and G. Lamont. *Evolutionary algorithms for solving multiobjective problems.* Kluwer Academic Publischers, 2002.

[37] D. Corne, N. Jerram, J. Knowles, and J. Oates. Pesa ii: Region-based selection in evolutionary multiobjective optimization. *Proc. of the Genetic and Evolutionary Computation Conference*, pages 283–290, 2001.

[38] D. Corne, J. Knowles, and M. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. *Lecture Notes in Computer Science*, 1917:839–848, 2000.

[39] Lawrence Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[40] K. Deb. Evolutionary algorithms for multi-criterion optimization in engineering design. In Kaisa Miettinen, Marko M. Mäkelä, Pekka Neittaanmäki, and Jacques Periaux, editors, *Evolutionary Algorithms in Engineering and Computer Science*, pages 135–161. John Wiley & Sons, Ltd, Chichester, UK, 1999.

[41] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 2001.

[42] K. Deb and R. Agrabal. Simulated binary crossover for continuous search space. *Complex System*, 9:115–148, 1995.

[43] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. *Parallel Problem Solving from Nature - PPSN VI*, 2000.

[44] K. Deb and D. Goldberg. An investigation of niche and species formation in genetic function optimization. *Proc. of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[45] K. Deb, M. Mohan, and S. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread Pareto-optimal solutions. Technical Report 2003002, KanGAL, 2003.

[46] Kalyanmoy Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[47] H. Ding, L. Benyoucef, and X. Xie. A simulation-based multi-objective genetic approach for networked enterprises optimization. *Engineering applications of artificial intelligence*, En Prensa, 2006.

[48] H. Ehrlich, M. Kühne, and J. Jäkel. Development of a fuzzy control system for greenhouses. *Acta Horticulturae*, 406:463–470, 1996.

[49] T. Erlebach, H. Kellerer, and U. Pferschy. Approximating multi-objective knapsack problems. *Lecture Notes in Computer Science*, 2125:210–221, 2001.

[50] L. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. *Foundations of Genetic Algorithms*, pages 265–283, 1991.

[51] K. Fedra, G. Van Straten, and M.B. Beck. Uncertainty and arbitrariness in ecosystems modelling: A lake modelling example. *Ecological Modelling*, 13:87–103, 1981.

[52] L. Ferariu and t. Marcu. Evolutionary design of dynamic neural networks applied to system identification. *Proceedings of the 15th IFAC World Congress*, 2002.

[53] T.C. Fogarty. Varying the probability of mutation in the genetic algorithm. *Proc. of Third International Conference on Genetic Algorithms*, pages 104–109, 1989.

[54] D.B. Fogel. *Evolutionary Computation. The Fossil Record. Selected Readings on the History of Evolutionary Algorithms.* The Institute of Electrical and Electronic Engineers, 1998.

[55] E. Fogel and F. Huang. On the value of information in system identification-bounded noise case. *Automatica*, 18(12):229–238, 1982.

[56] L.J. Fogel, A.J. Owens, and M.J. Walsh. *Artificial Intelligence Thorough Simulated Evolution.* Wiley Publishing, 1966.

[57] C. Fonseca. *Multiobjective genetic algorithms with application to control engineering problems.* PhD thesis, Dpt. of Automatic Control and Systems Engineering. University of Sheffield, 1995.

[58] C. Fonseca and P. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. *Proc. of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[59] M. Fourman. Compaction of symbolic layout using genetic algorithms. *Genetic Algorithms and their applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153, 1985.

[60] A. Garulli, B. Kacewicz, A. Vicino, and G. Zappa. Error bounds for conditional algorithms in restricted complexity set membership identification. *IEEE Transaction on Automatic Control*, 45(1):160–164, January 2000.

[61] L. Giarré and G. Zappa. Approximation of feasible parameter set in worst case identification of block-oriented nonlinear models. *12th IFAC Symposium on System Identification*, pages 869–874, 2002.

[62] D. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. *Proc. of the Third International Conference on Genetic Algorithms*, pages 41–49, 1987.

[63] D.E. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesley, 1989.

[64] D.E. Goldberg. Sizing population for serial and parallel genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms.*, Foundations of Genetic Algorithms I:70–79, 1989.

[65] E. Gomez-Ramírez, A.S. Poznyak, and R. Lozano. Polynomial artificial neural network and genetic algorithm for the identification and control of nonlinear systems. *Proceedings of the 15th IFAC World Congress*, 2002.

[66] P. Gray, W. Hart, L Painton, C. Phillips, M. Trahan, and J. Wagner. A survey of global optimization methods. Technical report, Sandia National Laboratories. Albuquerque., 1997.

[67] G. Greenwood, G. Fogel, and M. Ciobanu. Emphasizing extinction in evolutionary programming. *Proc. of the Congress of Evolutionary Computation*, 1:666–671, 1999.

[68] J. Grefenstette. Genetic algorithms for changing environments. *Proc. of Parallel Problem Solving from Nature II*, pages 137–144, 1992.

[69] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE transactions on Systems, Man and Cybernetics*, 16(1):122–128, 1986.

[70] S. Helbig and D. Pateva. On several concepts for ϵ-efficiency. *OR Spectrum*, 16(3):179–186, 1994.

[71] J.M. Herrero. Aplicación de los algoritmos genéticos en la identificación y control de procesos. Technical report, Departamento de Ingeniería y Automática. U.P.V., 2001.

[72] J.M. Herrero, X. Blasco, C. Ramos and J.V. Salcedo. *Identificación Robusta en Modelos no Lineales. Determinación del FPS.* XI Congreso Latinoamericano de Control. (Informática 2004), 2004.

[73] J.M. Herrero, X. Blasco, M. Martínez, and C. Ramos. Nonlinear robust identification using multiobjective evolutionary algorithms. In *Lecture Notes in Computer Science*, volume 3562, pages 231–241. Springer-Verlag, 2005.

[74] J.M. Herrero, X. Blasco, M. Martínez, and C. Ramos. Nonlinear robust identification with ϵ-GA: FPS under several norms simultaneously. In *Lecture Notes in Computer Science*, volume 3512, pages 993–1001. Springer-Verlag, 2005.

[75] J.M. Herrero, X. Blasco, M. Martínez, and J. V. Salcedo. Optimal PID tuning with genetic algorithms for non linear process models. *15th World Congress IFAC 2002*, 2002.

[76] J.M. Herrero, X. Blasco, M. Martínez, and J. Sanchis. Identification of continuous processes parameters using genetic algorithms. *Proceedings of the 10th Mediterranean Conference on Control and Automation*, 2002.

[77] J.M. Herrero, X. Blasco, J.V. Salcedo, and C. Ramos. Membership-set estimation with genetic algorithms in nonlinear models. *In Proc. of the XV international Conference on Systems Science*, September 2004.

[78] J.M. Herrero, X. Blasco, J.V. Salcedo, and C. Ramos. Membership-set estimation with genetic algorithms in nonlinear models. *Systems Science*, 31:67–76, 2005.

[79] J.M. Herrero, M. Martínez, X. Blasco, C. Ramos, and S. Rodríguez. Identificación paramétrica del modelo no lineal de un invernadero mediante algoritmos genéticos. *Actas de las XXIV Jornadas de Automática*, 2003.

[80] A. Herreros. *Diseño de controladores robustos multiobjetivo por medio de algoritmos genéticos*. PhD thesis, Universidad de Valladolid. Departamento de ingeniría de sistemas y automática., 2000.

[81] A. Herreros, E. Baeyens, and J.R. Perán. Parameter estimation of vibratory systems. *Proceedings of the 10th Mediterranean Conference on Control and Automation*, 2002.

[82] F. Hoffmeister and Z. Michalewicz. Problem-independent handling of constraints by use of metric penalty functions. *Proc. of the Fifth Annual Conference on Evolutionary Programming*, pages 289–294, 1996.

[83] J.H. Holland. *Adaptation in natural and artificial systems*. Ann Arbor: The University of Michigan Press, 1975.

[84] A. Homaifar, S. Lai, and X. Qi. Constrained optimization via genetic algorithm. *Simulation*, 62(4):242–254, 1994.

[85] J. Horn, D. Goldberg, and K. Deb. Implicit niching in a learning classifier system: nature's way. *Evolutionary Computation*, 2(1):37–66, 1994.

[86] J. Horn, N. Nafpliotis, and D. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. *Proc. of the First IEEE Conference on Evolutionary Computation*, 1:82–87, 1994.

[87] P.J. Huber. *Robust statistics*. John Wiley & Sons, New York, 1981.

[88] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm and its application to flowship scheduling. *IEEE transactions on systems man and cybernetics*, 28(3):392–403, 1998.

[89] C.Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representation in genetic algorithms. *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 31–36, 1991.

[90] R. Johansson. *System modelling identification*. Prentice Hall, 1993.

[91] J. Joines and C. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with gas. *Proc. of the first IEEE Conference on Evolutionary Computation*, pages 579–584, 1994.

[92] O. Jolliet and B. Bailey. The effect of climate on tomato transpiration in greenhouses: measurements and models comparison. *Agric. Forest. Meteor.*, 58:43–62, 1992.

[93] A. K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[94] K. A. De Jong and W. M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. *Intelligence Journal*, 5(1):1–26, 1992.

[95] K.J. Keesman. Membership-set estimation using random scanning and principal. *Mathematics and Computers in Simulation*, 32:535–544, 1990.

[96] K.J. Keesman and R. Stappers. Nonlinear set-membership estimation: A support vector machine approach. *J. Inv. Ill-Posed Problems*, 12(1):27–41, 2004.

[97] J. Knowles and D. Corne. Local search, multiobjective optimization and the Pareto archived evolution strategy. *Proceedings of Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems,*, pages 209–216, 1999.

[98] J. Knowles and D. Corne. The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation. *Proceedings of the 1999 Congress on evolutionary computation*, pages 98–105, 1999.

[99] J.R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* The MIT Press, 1992.

[100] T. Krink, R. Thomsen, and P. Rickers. Applying self-organised criticality to evolutionary algorithms. *Parallel Problem Solving from Nature VI*, 1:375–384, 2000.

[101] A. Kuri and C. Villagas. A universal eclectic genetic algorithm for constrained optimization. *Proc. 6th European Congress on Intelligent Techniques and Soft Computing*, pages 518–522, 1998.

[102] E. Walter and L. Pronzalo. *Identification of parametric models from experimental data.* Springer, 1997.

[103] H. Lahanier, E. Walter, and R. Gomeni. OMNE: a new robust membership-set estimator for the parameters of nonlinear models. *Pharmacokinetics*, 15:101–107, 1987.

[104] M. Laumanns. *Analysis and applications of evolutionary multiobjective optimization algorithms.* PhD thesis, Swiss Federal institute of Technology, Zürich, Switzerland, 2003.

[105] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

[106] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 46–53, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 2000. IEEE Press.

[107] R. Linker, I. Seginer, and P.O. Gutman. Optimal CO2 control in a greenhouse modelled with neural networks. *Computers and electronics in agriculture*, 19:289–310, 1998.

[108] L. Ljung. *System Identification, Theory for the user (2nd ed.)*. Prentice-Hall, 1987.

[109] A.M. Lopez, H. Lopez, and L. Sanchez. Ga-p based modelling of nonlinear dynamic systems. *Proceedings of the 15th IFAC World Congress*, 2002.

[110] S. Mahfoud. Crowding and preselection revisited. *Parallel Problem Solving from Nature*, 2:27–36, 1992.

[111] Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, IlliGAL, 1995.

[112] M. Martínez. Estimación de parámetros en modelos de bloqueo de fármacos. Technical report, Departamento de Ingeniería de Sistemas y Automática. UPV, 2006.

[113] M. Martínez, X. Blasco, J.M. Herrero, C. Ramos, and J. Sanchis. Monitorización y control de procesos. Una visión teórico-práctica aplicada a invernaderos. *Revista Iberoamericana de Automática e Informática Industrial*, 2(4):5–24, 2005.

[114] M. Martínez, J. Sanchis, and X. Blasco. Identification multiobjective controller design handling human preferences. *Engineering Applications of Artificial Intelligence*, En Prensa, 2006.

[115] C.A. Mattson, A. Mullur, and A. Messac. Smart Pareto filter: Obtaining a minimal representation of multiobjective design space. *Engineering Optimization*, 36(6):721–740, 2004.

[116] O. Mengshoel and D. Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. *Proc. of the Genetic and Evolutionary Computation Conference*, 1:409–416, 1999.

[117] A. Messac, A. Ismail-Yahaya, and C.A. Mattson. The normalized normal constraint method for generating the Pareto frontier. *Structural and Multidisciplinary Optimization*, 25:86–98, 2003.

[118] K. Metselaar, K. Keesman, W. Van der werf, and H. Van Keulen. Uncertainty appraisal report. Technical report, SAFE project, 2004.

[119] E. Mezura. *Alternative techniques to handle constraints in evolutionary optimization*. PhD thesis, Centro de Investigaciones y de Estudios Avanzados del Instituto Politécnico Nacional, México City, 2004.

[120] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1996.

[121] Z. Michalewicz and N. Attia. Evolutionary optimization of constrained problems. *Proc. of the 3rd annual Conference on Evolutionary Programming*, pages 98–104, 1994.

[122] Z. Michalewicz and D. Fogel. *How to solve it: Modern Heuristics*. Springer, 2000.

[123] Z. Michalewicz and G. Nazhiyath. GENOCOP III: A coevolutionary algorithm for numerical optimization problems with nonlinear constraints. *Proc. of the 1995 IEEE Conference on Evolutionary Computation*, pages 647–651, 1995.

[124] K. Miettinen. *Nonlinear multiobjective optimization.* Kluwer Academic Publishers, 1998.

[125] M. Milanese and A. Vicino. Robust estimation and exact uncertainty intervals evaluation for nonlinear models. *System Modelling and Simulation*, pages 91–96, 1989.

[126] M. Milanese and A. Vicino. Optimal estimation theory for dynamic systems with set membership uncertainty: An overview. *Automatica*, 27(6):997–1009, 1991.

[127] S.H. Mo and J.P. Norton. Fast and robust algorithm for the intersection and approximation of n-dimensional polytopes. *Math. and Computers in Simulation*, 32:481–493, 1990.

[128] J.L. Monteith. *Principles of environmental physics. Contemporary biology.* Edward Arnold Ed., UK, 1973.

[129] H. Mullenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm: I. continuous parameter optimization. *Evolutionary computation*, 1(1):25–49, 1993.

[130] B. Nielsen and H. Madsen. Identification of transfer functions for control of greenhouse air temperature. *Journal of Agricultural Engineering Research*, 60:25–34, 1996.

[131] J.P. Norton. Identification and application of bounded-parameter models. *Automatica*, 23(4):497–507, 1987.

[132] J.P. Norton. Identification of parameter bounds for ARMAX models from records with bounded noise. *Internat. J. Control*, 45:375–390, 1987.

[133] J.P. Norton and S. Veres. Identification of nonlinear state-space models by deterministic search. *In Proc. of the IFAC Symposium on Identification and system parameter estimation*, 1:363–368, july 1991.

[134] D. Orvosh and L. Davis. Shall we repair?genetic algorithms, combinatorial optimization and feasibility constraints. *Proc. of the Fifth International Conference on Genetic Algorithms*, page 650, 1993.

[135] J. Paredis. Co-evolutionary constraint satisfaction. In Springer Verlang, editor, *Proc. of 3rd Conference on Parallel Problem Solving from Nature*, pages 46–55, 1994.

[136] D. Powell and M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. *Proc. of the Fifth International Conference on Genetic Algorithms*, pages 424–431, 1993.

[137] B. Ram, H. Gupta, P. Bandyopadhyay, K. Deb, and V. Adimurthy. Robust identification of aerospace propulsion parameters using optimization techniques based on evolutionary algorithms. Technical Report 2003005, KanGAL, 2003.

[138] I. Rechenberg. Cybernetic solution path of an experimental problem. ministry of aviation, royal aircraft establishment. 1965.

[139] W. Reinelt, A. Garulli, and L. Ljung. Comparing different approaches to model error modelling in robust identification. *Automatica*, 38(5):787–803, May 2002.

[140] F. Rodríguez. *Modelado y control jerárquico de crecimiento de cultivos en invernadero*. PhD thesis, Universidad de Almería, Almería, 2002.

[141] G. Rudolph and A. Agapie. Convergence properties of some multi-objective evolutionary algorithms. *Congress on Evolutionary Computation*, 2:1010–1016, 200.

[142] J. Schaffer. *Multiple objective optimization with vector evaluated genetic algorithms*. PhD thesis, Vanderbilt University, Nashville, Tennessee, 1984.

[143] J.D. Schaffer, R.A. Caruana, L. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proceedings if third international Conference on Genetic Algorithms*, 1989.

[144] M. Schoenauer and S. Xanthakis. Constrained ga optimization. *Proc. of the Fifth International Conference on Genetic Algorithms*, pages 573–580, 1993.

[145] I. Seginer. The Penman-Monteith evapotranspiration equation as an element in greenhouse ventilation design. *Journal of Agricultural Engineering Research*, 82(4):423–439, 2002.

[146] I. Seginer, T. Boulard, and J.B. Bailey. Neural network models of the greenhouse climate. *Journal of Agricultural Engineering Research*, 59(3):203–216, 1994.

[147] R. Sikora and M. Shaw. A double-layered learning approach to acquiring rules for classification: Integrating genetic algorithms with similarity-based learning. *ORSA Journal on Computing*, 6(2):174–187, 1994.

[148] N. Srinivas and K. Deb. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[149] C. Stanghellini and T. de Jong. A model of humidity and its applications in a greenhouse. *Agricultural and Forest Meteorology*, (76):129–148, 1995.

[150] C.F. Starmer. Characterizing activity-dependent processes with a piecewise exponencial model. *Biometrics*, 44:549–559, 1988.

[151] G. Syswerda. Uniform crossover in genetic algorithms. In J. David Shaffer, editor, *Proc. 3rd International Conf. on Genetic Algorithms*. Kaufmann Publishing, 1989.

[152] G. Toscano and Carlos A. Coello. The micro-genetic algorithm 2: Towards on-line adaptation in evolutionary. 2003.

[153] R. Ursem. Multinational evolutionary algorithms. *Proc. of the Congress of Evolutionary Computation*, 3:1633–1640, 1999.

[154] R. Ursem. Diversity-guided evolutionary algorithms. *Proc. of Parallel Problem Solving from Nature*, pages 462–471, 2002.

[155] Rasmus K. Ursem. *Models for Evolutionary Algorithms and Their Applications in System Identification and Control Optimization.* PhD thesis, University of Aarhus, 2003.

[156] K. Rodríguez Vázquez. *Multiobjective evolutionary algorithms in non-linear system identification.* PhD thesis, Department of Automatic Control & Systems Engineering, 1999.

[157] E. Walter and M. Kieffer. Interval analysis for guaranteed nonlinear parameter estimation. *In Proc. of the 13th IFAC Symposium on System Identification*, 2003.

[158] E. Walter and H. Piet-Lahanier. Exact recursive polyhedral description of the feasible parameter set for bounded-error models. *IEEE Trans. Automatic Control*, 34(8):911–915, 1989.

[159] E. Walter and H. Piet-Lahanier. Estimation of parameter bounds from bounded-error data: A survey. *Mathematics and computers in Simulation*, 32:449–468, 1990.

[160] E. Walter and H. Piet-Lahanier. Recursive robust minmax estimation for models linear in their parameters. *In Proc. of the IFAC Symposium on Identification and system parameter estimation*, 1:763–768, July 1991.

[161] A. Weltzel. Evaluation of the effectiveness of genetic algorithm in combinational optimization. University of Pittsburgh, 1983.

[162] D. Whitley. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121, 1989.

[163] J.J. Rey William. *Introduction to robust and quasi-robust statistical methods.* Springer, Berlin, 1987.

[164] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Instituto de Santa Fe, 1995.

[165] P.C. Young, A. Chotai, and W. Tych. Identification, estimation and true digital control of glasshouse system. *The Computerized Greenhouse*, pages 3–50, 1993.

[166] M. Zamora. *Un estudio de los sistemas termodinámicos*. Universidad de Sevilla, 1998.

[167] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing Journal*, 15(1):59–76, 1996.

[168] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, 1999.

[169] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: empirical results. *Evolutionary computation*, 8(2), 2000.

[170] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, 2001.