UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escuela Técnica Superior de Ingenieria Informática
Universidad Politécnica de Valencia

Desarrollo y Evaluación de Algoritmos Genéticos Multiobjetivo. Aplicación al problema del viajante

Trabajo de fin de grado

Grado de Ingenieria Informática

Autor: Corral Sastre, Antonio

Tutor: Barber Sanchís, Federico

Curso académico: 2017-2018

Resumen

En este TFG se plantea resolver el problema del viajante con múltiples objetivos a optimizar, como extensión del problema básico. Al respecto, se analizarán las propiedades del mismo, así como las referentes a los problemas multiobjetivo en general. Tras ello, se desarrollará un método metaheurístico que permita obtener soluciones factibles y optimizadas. Para este cometido, se prevén estudiar y aplicar diferentes aproximaciones de algoritmos genéticos multiobjetivo, analizando y comentando los rasgos generales correspondientes, sus ventajas e inconvenientes.
Finalmente, se implementarán los algoritmos analizados anteriormente, sobre los que aplicarán diversas métricas de evaluación a fin de comparar los diferentes resultados obtenidos. Con ello, se podrán determinar las variantes de algoritmos genéticos multiobjetivo más eficaces para resolver adecuadamente el problema del viajante con múltiples objetivos.

Palabras clave: algoritmo genético, optimización multiobjetivo, problema del viajante, NSGA, SPEA

Resum

En aquest TFG es planteja resoldre el problema del viatger amb diversos objetius a optimitzar, com extensió del problema básic. Al respecte, s'analitzaran les propietats del mateix, així com les referents als problemes multiobjectiu en general. Després, es desenvolupará un métode metaheurístic que permetrá obtindre solucions factibles i optimitzades. Per aconseguir-ho, es preveu estudiar i aplicar diferents aproximacions d'algorismes genétics multiobjectiu, analitzant y comentant les característiques generals corresponents, els seus avantatges e inconvenients.
Finalment, s'implementaran els algorismes analitzats anteriorment, sobre els que s'aplicaran diverses métriques d'evaluació a fi de comparar els diferents resultats obtinguts. Amb això, es podran determinar les variants d'algorismes genétics multiobjectiu més efectives per resoldre de forma satisfactoria el problema del viatger amb múltiples objectius.

Paraules clau: algorisme genétic, optimització multiobjectiu, problema del viatger, NSGA, SPEA

Abstract

In this project it is proposed to solve the traveler salesman problem with multiple objectives to be optimized, as an extension of the basic problem. In this regard, the properties of the same will be analyzed, as well referring to the multiobjective problems in general. After that, a metaheuristic method will be developed to obtain feasible and optimized solutions. For this purpose, it is planned to study and apply different approaches of multiobjective genetic algorithms, analyzing and commenting their general features, advantages and disadvantages.
Finally, the algorithms analyzed above will be implemented, on which various evaluation metrics will be applied in order to compare the different results obtained. With this, the most effective variants of multiobjective genetic algorithms can be determined to adequately solve the traveler salesman problem with multiple objectives.

Keywords: genetic algorithm, multiobjective optimization, traveler salesman problem, NSGA, SPEA

Lista de algoritmos

Lista de figuras

Lista de tablas

Índice

1. Introducción

Problemas de la vida real pueden tratarse como un problema de optimización de un único objetivo, que puede resolverse con algoritmos genéticos multiobjetivo, pero la mayoría de problemas de la vida real que se pueden extrapolar y tienen varios objetivos que satisfacer a pesar de que algunos pueden ser conflictivos entre ellos.

1.1 Motivación

La motivación de este proyecto radica en la búsqueda de métodos para solucionar problemas de optimización multiobjetivo de forma eficiente. Muchos de los problemas de la vida cotidiana están compuestos por varios objetivos los cuales pretendemos optimizar de la mejor forma posible, a pesar que normalmente la mejora de un objetivo suele significar el empeoramiento de otro. Este tipo de problemas es tal que afecta incluso a decisiones del día a día como que medio de transporte a escoger, el orden de las tareas a realizar en varios lugares durante el día, etc. Los ingenieros también se suelen encontrar con este tipo de problemas como logística, planificación, fabricación de circuitos electrónicos, centrales energéticas, etc. (*Warehouse location problem*, Locom; *Process scheduling*, Unllever; *Job shop scheduling*, Deer y Company, SAP, Volvo; *Turbine design*, Rolce Royce, Honda; *Portfolio optimization*, First Quadrant; *Cleaning team assignment*, Die Bahn; *Chip design*, Texas Instruments; *Roboter movement*, Honda; *Nuclear fuel reloading*, Siemens; ...).

Un ejemplo muy utilizado en la literatura de un problema de optimización multobjetivo consiste en la adquisición de un vehículo. No habría que guiarse por gustos ni guiarnos solo por el dinero del cual disponemos, sino que habría que analizar la durabilidad de cada componentes al igual que la calidad de los mismos y analizar todos estas variables en conjunto para poder determinar si estamos realizando la mejor compra respecto al resto de vehículos. En la siguiente figura se observa el ejemplo del vehículo donde se optimiza el confort y el precio.

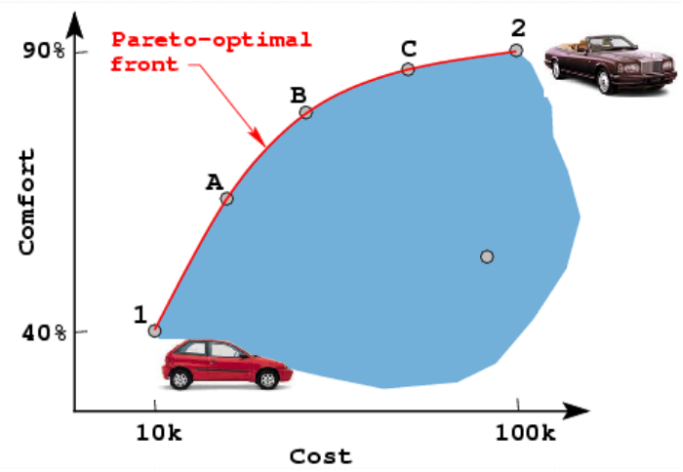


Figura 1: Problema biobjetivo del coche

Además nos encontramos aplicaciones como Google Maps donde para ir de un lugar a otro nos muestra varias opciones de caminos con diferentes medios de transporte, pero además de tener en cuenta factores como el combustible de los coches, se puede tener también en cuenta factores como el tiempo para llegar al destino, o posibles atascos según la hora que sea en un tramo del recorrido. El camino más corto (lo que *a priori* significaría un ahorro de combustible) puede que no sea siempre la mejor opción, ya que a lo mejor este camino es más accidentado que otro y posiblemente se tardaría más tiempo en recorrer que otro, lo que significaría al final un gasto mayor de combustible. Algunos factores, aunque independientes entre si, pueden ser proporcionales entre si (cuanto más corto sea el camino, *a priori* menos combustible se utiliza). La relación de los objetivos y variables de este tipo de problemas se analizará más adelante en cuanto introduzcamos los algoritmos genéticos multiobjetivo.

Hay algoritmos para problemas monobjetivo que son muy eficaces como el algoritmo SIMPLEX o el Algoritmo del Punto Interior, sin embargo para problemas multiobjetivo lo que se ha tendido a

hacer para resolverlos es combinar los diversos objetivos en un objetivo único que engloba todos ellos además de que la mayoría de ellos consisten en priorizar un objetivo y el resto se ponen como restricciones. Actualmente habrá medio centenar y un centenar de algoritmos que resuelvan problemas multiobjetivo, aunque la mayoría de ellos necesitan un punto inicial de búsqueda y suelen generar una única solución por ejecución (casi nunca este tipo de problemas tienen una única solución).

La evolución genética se han probado que es bastante satisfactoria a la hora de resolver problemas monobjetivo como multiobjetivo, debido a que podemos capturar un número de soluciones de forma concurrente de una simple pasada durante la ejecución del algoritmo, además de observar que del mecanismo el cual está inspirado ha llegado a crear vida con inteligencia.

1.2 Objetivos

Los objetivos a cumplir en el proyecto son los siguientes:

- Comprender la estructura básica de los algoritmos genéticos y determinar que parámetros y métodos hay que usar según el problema para aplicarlos de forma satisfactoria.
- Analizar los puntos claves e introducir los conceptos necesarios para poder solucionar correctamente un problema multiobjetivo.
- Explicación general de los estado del arte de los algoritmos genéticos multiobjetivo revisando como han ido cambiado hasta fecha de hoy y profundizar en aquellos que son los más utilizados.
- Implementar una librería básica para comparar en el problema del viajante el máximo número posible de estado del arte de los algoritmos genéticos multiobjetivo. Además, intentar que esta librería sea utilizable en el máximo número de problemas multiobjetivo.

1.3 Estructura de la memoria

La memoria se ha estructurado respectivamente en los siguientes apartados:

- Análisis de las características claves en problemas de optimización multiobjetivo para encontrar una buena solución, además de la clasificación de estos.
- Exposición del problema del viajante (problema a resolver), al que le aplicaremos los algoritmos genéticos multiobjetivo que comentaremos más adelante.
- Introducción y explicación de las diversas etapas que conforman un algoritmo genético monobjetivo.
- Explicación de los conceptos necesarios a tener en cuenta para los algoritmos genéticos multiobjetivo, al igual que los estado del arte de los algoritmos genéticos multiobjetivo tanto los de primera como los de segunda generación.
- Uso de varias librerías para testear los algoritmos genéticos multiobjetivo aplicados al problema del viajante.
- Conclusiones y posibles mejoras tanto con las librerías como en los algoritmos genéticos multiobjetivo implementados y usados en las librerías.

2. Problemas de optimización multiobjetivo

Los problemas de la vida real requieren de soluciones que cumplan al mismo tiempo más de un objetivos o criterios, los cuales en la mayoría de los casos suelen ser contradictorios. Cuando los objetivos de un problema se pueden combinar de forma adecuada, se puede realizar una optimización de un único objetivo, (minimización/maximización), sin embargo, lo usual es que no se conozca o pueda realizar la forma óptima de combinar los diferentes objetivos.

Además, en los problemas multiobjetivo con objetivos contradictorios entre sí no siempre existe una solución única que se puede considerar como óptima, sino que se consideran un conjunto de soluciones que reflejan el mejor compromiso entre los objetivos, ya que normalmente la mejora de un objetivo significa el empeoramiento de otro objetivo. Dicho conjunto es llamado conjunto Pareto óptimo, mientras que frente de Pareto se le llama a la imagen del conjunto Pareto óptimo en el espacio objetivo. Estos conceptos serán comentados más adelante.

Hay varias técnicas a utilizar para resolver problemas de optimización multiobjetivo. Se pueden utilizar algoritmos exactos o de fuerza bruta que nos van a garantizar la solución más óptima, sin embargo estos suelen ser demasiado costosos computacionalmente hablando. Las más utilizadas suelen ser bioinspiradas, basadas con heurísticas que abarcan inteligencia artificial, evolución biológica e inteligencia colectiva, y entran dentro del conjunto de EMO (*Evolutionary Multiobjective Optimization*). En los últimos tiempos estas técnicas, incluso las no basadas en algoritmos evolutivos han surgido como alternativas a los algoritmos genéticos multiobjetivo. Algunas de las técnicas más conocidas son:

- Algoritmos culturales.
- Algoritmos genéticos.
- Colonias de hormigas.
- Nubes de partículas (*Particle Swarm*).

Un problema de optimización multiobjetivo, también conocido por *Multiobjective Optimization Problem* (MOP), está compuesto por:

- n variables de decisión.
- k funciones objetivo.
- m restricciones.

Matemáticamente puede expresarse de la siguiente manera:

$$\boxed{\begin{array}{ll} \text{Optimizar} & y = f(x) = (f_1(x), f_2(x), ..., f_k(x)) \\ \text{sujeto a} & e(x) = (e_1(x), e_2(x), ..., e_m(x)) \geq 0 \\ \text{donde} & x = (x_1, x_2, ..., x_n) \in X \subseteq \mathbb{R}^n \\ & y = (y_1, y_2, ..., y_k) \in Y \subseteq \mathbb{R}^k \end{array}} \quad (1)$$

donde

- x es el vector de decisión n-dimensional.
- y el vector objetivo k-dimensional.
- El espacio de decisión se le denomina por X.
- El espacio objetivo es conocido como Y.
- El conjunto de restricciones dado por la función $e(x)$ define la región de factibilidad $X_f \subseteq \mathbb{R}^n$ y cualquier punto $t \in X_f$ contenido en esa región es una solución factible.

Optimizar, dependiendo del problema, consiste en minimizar o maximizar la función $f(x)$. En la optimización monoobjetivo, existe una función que evalúa la solución X, conocida como $f : X \rightarrow Y$. El valor de Y se le conoce como el valor de aptitud o de *fitness*.

Si la función de evaluación f está compuesta de múltiples dimensiones, entonces $Y \subseteq \mathbb{R}^k$ con $k > 1$. En este caso, determinar si una solución es más óptima que otra puede ser una tarea compleja. Para solucionar este problema, se utiliza un concepto ampliamente difundido denominado dominancia de Pareto.

En este tipo de problemas el espacio de búsqueda se encuentra parcialmente ordenado y dos soluciones pueden ser equivalentes entre sí, siendo muy poco común que un único objetivo optimice al mismo tiempo todos los objetivos.

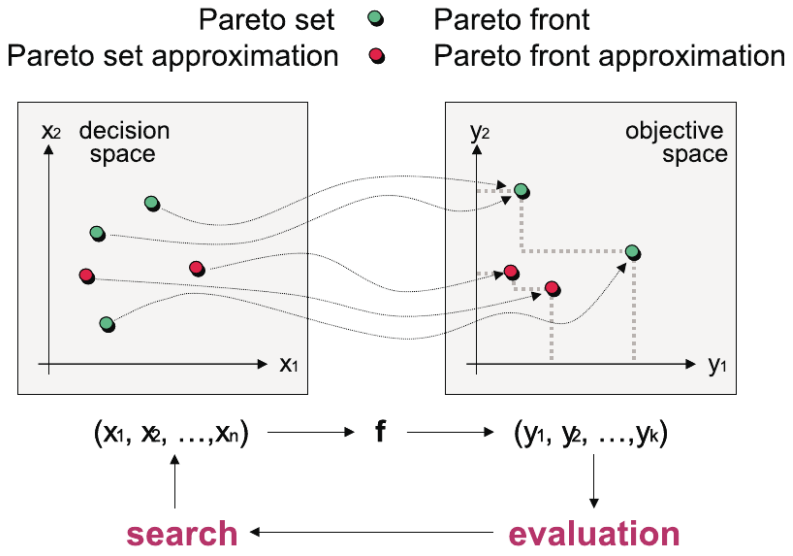


Figura 2: Espacio de decisión/espacio objetivo

El conjunto de restricciones $e(x) \geq 0$ determina el conjunto de soluciones factibles X_j y su correspondiente conjunto de vectores objetivo factibles Y_f que representan la frontera entre una solución factible y una la cual no lo es.

Por último, hay que tener en cuenta que cada función objetivo esta compuesta por una única variable independiente del resto de variables del resto de objetivos a optimizar, por lo que se puede deducir que el maximizar una función objetivo es lo mismo que minimizar con valor negativo la función objetivo en cuestión.

2.1 Búsqueda y toma de decisiones

Para resolver un problema de optimización multiobjetivo se han de tener en cuenta dos aspectos del problema:

- Búsqueda: Aborda el proceso de optimización en el cual se explora el espacio de soluciones factibles para buscar soluciones Pareto óptimas, aunque puede que se encuentren soluciones factibles no Pareto óptimas debido a que el espacio de búsqueda puede ser grande y complejo.
- Toma de decisiones: Consiste en seleccionar un conjunto de soluciones que se utilizarán para pasos posteriores. Este paso normalmente lo realiza un ser humano y tiene que ver con cuestiones relacionadas con las condiciones del problema y los recursos disponibles.

2.2 Clasificación

Dependiendo de como se combinan ambas fases para la resolución de un problema multiobjetivo, los métodos pueden clasificarse en diversas categorías:

- Toma de decisión previa a la búsqueda: Consiste en decidir y luego buscar. Los objetivos del problema de optimización multiobjetivo se combinan en un único objetivo que incluye información de preferencia obtenida de la toma de decisiones. La mayoría de las técnicas tradicionales para resolver problemas de optimización multiobjetivo utilizan esta estrategia. Tienen como ventaja de que la función objetivo combinada puede aplicarse a cualquiera de los métodos de optimización monobjetivo sin ningún tipo de modificación, sin embargo, la combinación de las funciones objetivo en una sola requiere conocer las características del problema, cosa que en ocasiones no se da. También son conocidos como métodos de toma de decisión *a priori*.

- Búsqueda previa a la toma de decisión: Consiste en buscar y luego decidir. Se resuelve el problema sin tener que tener en consideración ninguna variable del mismo. El agente que toma las decisiones tiene que escoger en cada iteración los individuos que el crea que son los más convenientes. A éstos métodos también se les conoce como de toma de decisión *a posteriori*.

- Toma de decisión durante la búsqueda: también conocidos como métodos interactivos o progresivos, consiste en buscar al mismo tiempo que se decide. Quien toma las decisiones puede establecer las preferencias de modo interactivo durante la ejecución del problema de un modo guiado. Luego de llevarse a cabo la resolución del problema, se suelen analizar los conflictos existentes entre los objetivos a optimizar para especificar sus preferencias o compromisos. Junto con los métodos de toma de decisión *a posteriori* tiene la dificultad de presentar al agente interactivo que toma las decisiones las diferencias entre las soluciones de un problema de optimización de muchos objetivos.

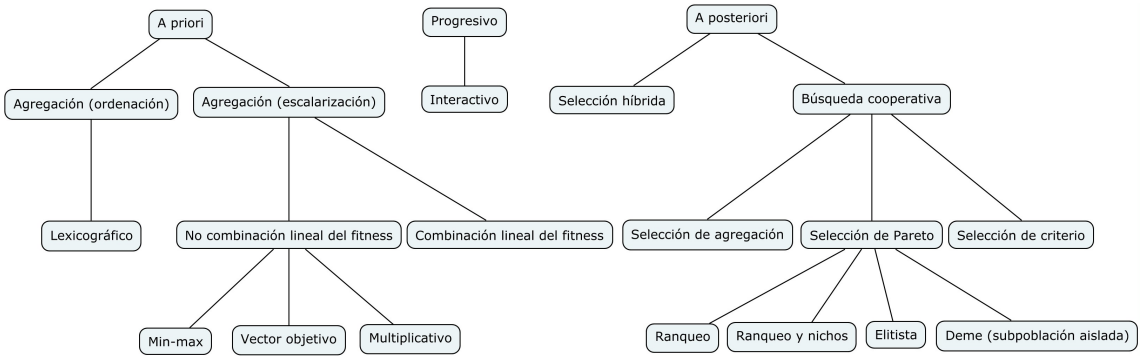
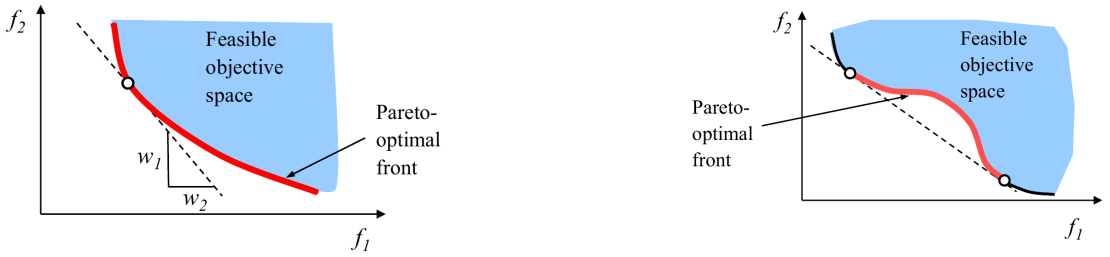


Figura 3: Clasificación de problemas de optimización multiobjetivo
Los algoritmos genéticos entran dentro del apartado de '*a posteriori*' y los métodos clásicos en el apartado de '*a priori*'

2.3 Métodos clásicos

Algunas de estas técnicas fueron aprovechadas para realizar los primeros algoritmos genéticos multiobjetivo o ser aprovechadas por el MOEA/D. Como las soluciones pueden ser vistas como puntos en el espacio de búsqueda, se pueden realizar a través de otros métodos, como el escalado de funciones, método de las restricciones o la programación en base a objetivos. Estos métodos siempre tienen que trabajar como mínimo con una solución candidata.

2.3.1 Agregación de funciones



(a) Frontera de Pareto convexa

(b) Frontera de Pareto cóncava

Figura 4: Frente convexo/frente cóncavo

También conocido como aproximación de la suma de pesos (Cohon, 1983; Miettinen, 1999) es probablemente, junto con el método de búsqueda aleatoria de soluciones, la técnica clásica más simple. Las funciones de los múltiples objetivos son combinados juntas en una única función Z, es decir, consiste en combinar todas las funciones objetivo $f_i(x)$ en una función única $F(f_1(x), ..., f_k(x))$.

La función más utilizada es la combinación lineal de los objetivos de la siguiente ecuación

$$\boxed{F = \sum_{i=1}^{k} \omega_i f_i(x)} \tag{2}$$

donde ω_i son las importancias de cada función objetivo, siendo común que sean normalizadas, tal que:

$$\boxed{\sum_{i=1}^{k} \omega_i = 1} \tag{3}$$

El vector de pesos tiene la función de asignar la preferencia de cada objetivo modificando su correspondiente peso. Matemáticamente, una solución obtenida con el mismo peso para cada objetivo suele dar menos conflictos en los objetivos. En la mayoría de casos, cada objetivo es optimizado individualmente y todos los valores objetivos de la función se calculan en cada solución individual óptima. Luego dependiendo de la importancia de los objetivos se escoge un vector de pesos acorde con los resultados de cada optimización individual. Las ventajas de usar esta técnica es que es fácil de implementar, que se puede escoger un objetivo y a través de este controlar el resto de objetivos y la solución óptima normalmente suele ser una solución Pareto óptima, sin embargo, tiene como desventajas que no funciona si la frontera de Pareto es cóncava, además de que el enfoque es muy sensible a la especificación de los pesos y solo se obtiene una solución en cada iteración del algoritmo. Aproximaciones como ε solucionan estas desventajas.

2.3.2 Orden lexicográfico

En este método, el agente de decisión es preguntado para ranquear los objetivos en orden de importancia. La óptima solución es obtenida minimizando (o maximizando según sea el caso) la funciones objetivo en secuencia, empezando por el más importante y procediendo de acuerdo al orden de importancia asignado a los objetivos. También es posible seleccionar aleatoriamente un objetivo a optimizar en cada generación si la prioridad es desconocida.
Seleccionando aleatoriamente un objetivo es equivalente a combinar los objetivos por pesos. Sin embargo, el uso de la selección por torneo en esta aproximación (Fourman) marca una importante diferencia respecto a otras técnicas como VEGA. Esto es debido a que la comparación de parejas por selección por torneo hace la información escalada negligible, por lo tanto esta aproximación puede compensar la representación de superficies cóncavas, aunque realmente depende de la distribución de la población y del problema en si. Las técnicas lexicográficas no han encontrado el favor de los investigadores de los MOEA, ya que solo hay unas pocas implementaciones reportadas en la literatura especializada. Esto se debe a que la debilidad de este enfoque es que tiende a favorecer objetivos más certeros cuando muchos están presentes en el problema, debido a la aleatoriedad

envuelta en el proceso. Esto tiene la indeseable consecuencia de hacer converger la población a una parte particular del frente de Pareto en vez de delinearlo completamente (explora el espacio objetivo de forma no equitativa). Su empeño se puede ver afectado por este orden, por lo tanto no es adecuado si se tratan muchos objetivos. La principal ventaja de esta aproximación es la simplicidad y eficiencia computacional. Estas dos propiedades hace que sea altamente competitivo con otras aproximaciones no Pareto como la suma de pesos por objetivo o VEGA. Las técnicas lexicográficas suelen ir mejor solo cuando la importancia de cada objetivo (en comparación con los otros) es de sobra conocido. Todas las soluciones reportadas por este método son Pareto óptimas, pero, si al hacerlo estamos optimizando un objetivo a costa de otros, parece más apropiado utilizar un algoritmo genético monobjetivo.

2.3.3 Método ε-constraint

También conocido como método de la perturbación ϵ (Haimes; Lasdon & Wismer, 1971). Los métodos ϵ-constante están basadas en seleccionar una función objetivo primaria y luego delimitar las otras (restricciones) con una ϵ-constante permitida (debe conocerse *a priori*). Luego, las ϵ-constante se modifican para generar otro punto en el frente de Pareto (fenotipo) y se obtienen elementos en el conjunto óptimo de Pareto (genotipo). Es un método fácil de implementar y aplicable tanto a problemas convexos como no convexos, pero requiere una alto esfuerzo de computación para generar el frente de Pareto conocido, al igual que el vector ϵ debe ser escogido con cuidado cogiendo valores entre el mínimo o máximo de cada función objetivo y los puntos del frente de Pareto no se suelen distribuir uniformemente.

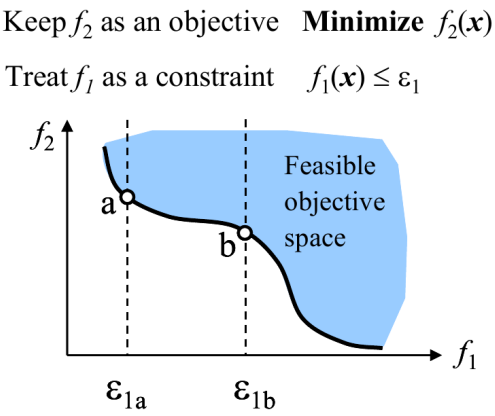


Figura 5: Método ϵ-constante

2.3.4 Vector objetivo

Se define un conjunto de metas que se desea alcanzar para cada función objetivo. El algoritmo minimizará la diferencia entre la solución y esas metas. También se pueden usar estrategias de agregación, por ejemplo min-max, aunque pueden generarse partes cóncavas del frente de Pareto. Las ventajas que tiene este método es que es fácil de implementar, teniendo como desventajas que la definición de objetivos puede tener costo computacional extra, al igual que puede producirse un desvió de la presión selectiva por la agregación de objetivos, además se tiene que tener en cuenta que deben cumplirse los objetivos para garantizar soluciones en el conjunto de Pareto.

2.3.5 Funciones de las distancias

En este método (Miettinen, 1999), se debe escoger un vector $\bar{y}$. La función objetivo derivada de múltiples objetivos es:

$$Z = \left[\sum_{i=1}^{N} \mid f_i(x) - \bar{y}_i \mid^r\right]^{1/r}, \qquad 1 \leq r \leq \infty, \tag{4}$$

donde $x \in X$ (región factible). Normalmente se suele escoger una métrica euclídea $r = 2$, con $\bar{y}$ como objetivo óptimo (Hans, 1998). Es importante notar que la solución obtenida depende del

vector $\bar{y}$, por lo que una selección arbitraria del vector no es muy recomendable ya que puede llevar a una solución no Pareto óptima. Como la solución no está garantizada, se deben conocer los objetivos a optimizar para escoger un vector apropiado. De esta forma, este método trabaja como una técnica de alcanzar objetivos impuestos por el vector objetivo $\bar{y}$. Este método es similar al método de la asignación de pesos a los objetivos. La única diferencia es que en este método el valor que se quiere alcanzar de cada función objetivo es necesaria mientras que en el método previo es requerida la importancia de cada objetivo.

También se puede combinar con pesos de la siguiente manera (se conoce como método de métrica de pesos Tchebycheff). Incluso está métrica se puede usar excluyendo los pesos

$$\boxed{Z = \left[\sum_{i=1}^{N} w_i |f_i(x) - \bar{y}_i|^r\right]^{1/r}, \qquad 1 \leq r \leq \infty,} \tag{5}$$

Las ventajas de este método es que garantiza todas las soluciones Pareto-óptimas con una solución ideal $\bar{y}$. Como desventajas encontramos que requiere con conocimiento del máximo y mínimo de los valores objetivo, requiere $\bar{y}$ que puede ser encontrado independientemente optimizando cada función objetivo. Para poblaciones pequeñas, no todas las soluciones Pareto óptimas son obtenidas, mientras que al aumentar la población, el problema cada vez se vuelve menos diferenciable. Además la función de agregación no es suave para MOP continuos.

2.3.6 Formulación Min-Max

Este método sigue un principio diferente al resto de los métodos expuestos anteriormente. Este método intenta minimizar las desviaciones relativas de cada función objetivo respecto al óptimo individual, es decir, intenta reducir la conflictividad entre objetivos. Para un problema de minimización, el correspondiente problema min-max se formula de la siguiente manera:

$$\boxed{\text{minimizar } F(x) = max\left[Z_j(x)\right], \quad j = 1, 2, ..., N,} \tag{6}$$

donde $x \in X$ (la región factible) y Z_j es calculado para el valor óptimo objetivo no negativo $\bar{f}_j > 0$ como sigue:

$$\boxed{Z_j(x) = \frac{f_j - \bar{f}_j}{\bar{f}_j}, \quad j = 1, 2, ..., N,} \tag{7}$$

Este método puede conseguir la mejor solución cuando los objetivos con la misma prioridad son requeridos para ser optimizados. Sin embargo, la prioridad de cada objetivo puede variar al introducir pesos dimensionales en la fórmula. Esto puede ser modificado como una técnica de alcanzar objetivos impuestos por el vector objetivo, como la formulación min-max.

3. Problema del viajante

El problema del viajante (*Traveller Salesman Problem*: TSP) fue formulado por primera vez en 1930 y trata de dar solución a la siguiente cuestión: dada una lista de ciudades y las distancias entre ellas, cuál es el camino más corto (minimización) que visita cada ciudad solo una vez y al alcanzar la última ciudad regresa a la ciudad de inicio. El problema suele ser modelado como un grafo ponderado (normalmente se trata de un grafo completo donde cada par de vértices está conectado por una arista) de manera que:

- Los vértices del grafo se corresponden con las ciudades.
- Los caminos entre ciudades se corresponden con las aristas.
- Los pesos de las aristas se corresponden con las distancias de los caminos.

Dependiendo del tipo de grafo que representa el problema nos podemos encontrar:

- Simétrico, donde la distancia entre un par de ciudades es la misma independientemente de la ciudad de inicio, formando un grafo no dirigido. Esta variante reduce a la mitad el número de soluciones posibles.
- En el TSP asimétrico pueden no existir caminos en ambas direcciones o las distancias pueden ser diferentes, formando un grafo dirigido. Casos en los que se da este tipo de TSP son cuando se consideran accidentes de tráfico, calles de un solo sentido o tarifas aéreas para con distintos costes de partida.

Para el conjunto de ciudades $0, ..., n$, si existe el camino para ir de la ciudad i a la ciudad j, x_{ij} es igual a 1, y 0 en otro caso. Sean u_i para $i = 1, ..., n$ variables artificiales para satisfacer las restricciones del problema y sea c_{ij} la distancia entre la ciudad i a la ciudad j, entonces el modelo de programación lineal en enteros puede ser descrito de la siguiente manera:

$$
\begin{array}{ll}
\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} & \\
0 \leq x_{ij} \leq 1 & i, j = 0, ...n \\
x_{ij} \text{ integer} & i, j = 0, ...n \\
\sum_{i=0, i \neq j}^{n} x_{ij} = 1 & j = 0, ...n \\
\sum_{j=0, j \neq i}^{n} x_{ij} = 1 & i = 0, ...n \\
u_i - u_j + n x_{ij} \leq n - 1 & 1 < i \neq j \leq n
\end{array}
\tag{8}
$$

El primer conjunto de igualdades asegura que cada ciudad de salida llegue exactamente a una ciudad, mientras que el segundo conjunto de igualdades aseguran que desde cada ciudad $1, ..., n$ se salga exactamente hacia una ciudad (ambas restricciones también implican que exista exactamente una salida desde la ciudad 0).
La última restricción sirve para garantizar las siguientes condiciones:

- La primera es que un único camino recorra todas las ciudades.
- La segunda que para cada recorrido haya valores u_i que satisfazcan las restricciones del problema (además x_{ij} debe de ser 0).

3.1 Complejidad computacional

En el problema se presentan N! rutas posibles, aunque se puede simplificar ya que dada una ruta no importa la ciudad de inicio y esto reduce el número de rutas a examinar en un factor N quedando (N-1)!. Además, en caso de que no nos importe la dirección en la que se desplace el viajante, el número de rutas a examinar se reduce de nuevo a (N-1)!/2 rutas posibles.

Para un TSP de 4 ciudades hay (4-1)!/2=3 rutas diferentes y para esta instancia del problema se puede resolver fácilmente con lápiz y papel. Por cada ciudad nueva que incorporemos, el número de rutas se multiplica por factor N y por lo tanto crece de forma factorial. Debido a este crecimiento factorial del problema, este pertenece a la clase de problemas NP-completos. Este es un problema NP-duro, un tipo de problemas muy importante en la ciencia de la computación. Los intentos de suavizar restricciones del problema no varia el hecho que el problema continue siendo NP-duro y continue perteneciendo a la clase de los NP-completos. En el peor caso, el tiempo de ejecución para cualquier algoritmo que resuelva un problema NP-duro seguramente aumente de forma exponencial (en nuestro problema aumenta exponencialmente respecto al número de ciudades). Los métodos para atacar un problema NP-duro como el problema del viajante son las siguientes:

- Formular algoritmos para encontrar soluciones exactas (estos trabajan más rápidos en problemas con dimensiones pequeñas).
- Formular algoritmos heurísticos o "subóptimos" (por ejemplo: algoritmos que den aparentemente o probablemente buenas soluciones, pero no devuelven el óptimo).
- Encontrar los casos especiales para el problema ("subproblemas") para los cuales heurísticas mejores o algoritmos exactos son posibles.

3.2 Algoritmos exactos/heurísticos

Aunque el problema es computacionalmente complejo, son conocidas una gran cantidad de heurísticas y métodos exactos que nos permiten resolver el problema con hasta miles de ciudades. Los más conocidas son:

- El modo más obvio para solucionar este problema es intentar todas las permutaciones (combinaciones ordenadas) y ver cuál de estas es la menor (utilizando una búsqueda de fuerza bruta). El tiempo de ejecución es de factor polinómico de orden $O(n!)$ donde n se corresponde al número de ciudades. Esta manera de solucionar el problema es inútil a partir de 10-15 ciudades.
- Uno de las mejores algoritmos para resolver el problema se basa en la programación dinámica que resuelve el problema en $O(n^2 2^n)$ (algoritmo Held–Karp). A partir de este punto, mejorar el coste temporal del algoritmo es demasiado difícil de encontrar. Por ejemplo, no ha sido demostrado si existe un algoritmo para el TSP que tenga un coste temporal de $O(1{,}9999^n)$.
- El algoritmo del vecino más próximo permite al viajante elegir la ciudad no visitada más cercana como próximo movimiento, devolviendo un camino corto en poco tiempo, sin embargo, dependiendo de la distribución de las ciudades, este algoritmo puede devolver un mal camino. Esto sucede tanto para TSP simétricos como asimétricos. Una variación de este algoritmo consiste en el operador de fragmento más cercano, la cual conecta un grupo (fragmento) de ciudades no visitadas más cercanas y puede encontrar la ruta más corta mediante iteraciones sucesivas. Este operador también puede ser aplicado también en la obtención de soluciones iniciales para el algoritmo del vecino más próximo para además ser mejorado mediante un modelo elitista.
- *Match Twice and Stitch.*
- Intercambio par a par.
- Heurística k-opt o heurística Lin-Kernighan.
- Heurística V-opt.
- Mejoras aleatorias.
- Optimización por colonia de hormigas.

Nosotros nos centraremos como método heurístico en la aplicación de algoritmos genéticos para resolver el problema.

El TSP despierta interés entre los investigadores debido es un problema muy sencillo de enunciar, pero difícil de resolver. Es aplicable a una gran variedad de problemas de planificación y

los nuevos métodos de optimización combinatoria son aplicados al TSP con la finalidad de tener un conocimiento más profundo del problema a resolver. Debido a estas aplicaciones, le ha hecho convertirse en una especie de problema test. Son numerosas las aproximaciones heurísticas bioinspiradas que se han desarrollado para el problema del viajante:

- La mayoría de ellos se describen en Lawler y col. (1985).
- La primera aproximación al problema a partir de algoritmos genéticos la efectuó Brady (1985). Su intento fue seguido por Grefenstette y col. (1985), Goldberg y Lingle (1985), Oliver y col. (1987), etc.
- También han sido aplicadas otras técnicas de algoritmos evolutivos, véanse por ejemplo, Fogel (1988), Banzhaf (1990) y Ambati y col. (1991).

4. Algoritmos genéticos monobjetivo

También conocidos como algoritmos genéticos simples, son técnicas utilizadas para resolver problemas de optimización y búsqueda. Están inspirados en el proceso genético de los seres/organismos vivos donde al largo de las generaciones los individuos evolucionan de acuerdo con los principios de la selección natural postulados por Darwin (1859).

Evolución natural	**Algoritmos evolutivos**
individuo	solución potencial
entorno	problema
fitness/como de bien está adaptado	coste/calidad de la solución
supervivencia del más adaptado	buenas soluciones son mantenidas
mutación	perturbaciones pequeñas/aleatorias
cruce	recombinación de soluciones parciales

Cuadro 1: Relación selección natural-algoritmos evolutivos

```
begin
    Conseguir una población inicial;
    while condicion_parada == False do
        Seleccionar individuos para el cruce;
        Cruzar con cierta probabilidad dos individuos obteniendo descendientes;
        Mutar descendientes con cierta probabilidad;
        Insertar descendientes mutados/quitar individuos de la población;
    end
end
```

Algoritmo 1: Algoritmo genético monobjetivo

Un algoritmo genético puede definirse como un proceso que actúa sobre los individuos candidatos a soluciones a través de dos pasos:

- La selección, que trata de imitar la competición por los recursos que existen en la naturaleza y la reproducción de los seres vivos.
- La variación, que trata de imitar la capacidad de creación de nuevas formas de vida a través de la recombinación y la mutación.

Además se ha de indicar una condición de parada del algoritmo. La evolución de dichas soluciones hacia el valor óptimo del problema depende en gran medida de dos cosas:

- Una adecuada codificación de las soluciones.
- Elección de una función objetivo adecuada.

Mediante el teorema de esquemas y los teoremas de convergencia se puede demostrar que los algoritmos genéticos funcionan (convergen cada vez más hacia una solución mejor). Para más información acerca de estos teoremas, hay que referirse a [ISG18].

Las estrategias evolutivas adaptadas para los algoritmos genéticos son:

- $(1 + 1)$-ES
- (μ, λ)-ES
- $(\mu/\mu_I, \lambda)$-ES

Además de ser utilizadas en otros campos como inteligencia artificial, hay técnicas híbridas que han sido aceptadas para los algoritmos genéticos:

- Búsqueda local genética.
- Adaptación de parámetros.

4.1 Codificación

Los individuos (posibles soluciones al problema, denominadas cromosomas) pueden representarse como un conjunto de componentes (denominados genes), los cuales en grupo conforman una ristra de valores. El alfabeto más utilizado para representar los individuos suele ser el alfabeto binario ($\{0, 1\}$) ya que la mayor parte de los conceptos teóricos realizados en los algoritmos genéticos han sido realizados con este alfabeto. Sin embargo, el alfabeto binario para el TSP no es la representación más simple, representativa o fácil de interpretar, además que al aplicar los operadores de cruce y mutación más comunes, la mayoría de las veces se tienen que reparar los individuos descendientes. Las representaciones más utilizadas para el problema del viajante son las siguientes:

- Representación basada en la trayectoria: El *path* es un cromosoma de longitud n ciudades que hay que recorrer que en el caso del problema del viajante consiste en una ristra de valores donde cada valor identifica a una ciudad y el siguiente valor representa que el viajante va de esa ciudad del valor anterior a la siguiente ciudad. Esta codificación suele simplificar los operadores de cruce y mutación, aunque los operadores clásicos no suelen dar un óptimo rendimiento en el algoritmo genético, por lo que más adelante se explicarán operadores un poco más complejos pero basados en los operadores clásicos que están probados que dan buen rendimiento en el problema TSP. Esta representación también conocida como gira (en programación se conoce como permutación). Por ejemplo, la gira $3-2-4-1-7-5-8-6$ se representará como (3 2 4 1 7 5 8 6).
- Binaria.
- Matricial.
- Ordinal.
- Lista de adyacencia.

4.2 Población

Se debe generar la población inicial donde cada individuo debe estar constituido por los valores del alfabeto con probabilidad uniforme. En principio estas soluciones pueden ser válidas o no, pero hay que tener en cuenta que la salida del algoritmo genético siempre tiene que devolver una solución válida. Estas se pueden generar:

- Aleatoriamente
- A partir de los resultados de una heurística previa, para que la población inicial este compuesta por soluciones válidas y cuyo valor *fitness* se acerque al valor *fitness* de la solución óptima, con la esperanza de que al aplicar el algoritmo genético a esa población se genere la solución óptima. Los pocos estudios que analizan la aplicación de una heurística previa para seleccionar la población inicial confirman que puede hacer que el algoritmo genético converja más rápido hacia soluciones óptimas, aunque esto conlleva algunos inconvenientes como:
  - Posible pérdida de la diversidad genética
  - Posible preferencia no recuperable

Algo que puede cuestionarse es el tamaño ideal de la población. Las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que trabajar con poblaciones de gran tamaño pueden llevar a un costo computacional elevado, sobretodo a la hora de calcular el valor de aptitud de los individuos de la población. Goldberg (1989) efectuó un estudio teórico de codificación binaria demostrando que el tamaño de la población crece exponencialmente con el tamaño del individuo, mientras que Alander (1992) demostró que un tamaño de población comprendida entre l y $2l$ es suficiente para atacar los problemas de forma efectiva.

4.3 Fitness

En este paso se evalúa cada individuo mediante la función *fitness* o aptitud a través de la función objetivo del problema. Las claves para construir una buena función objetivo que nos proporcione el valor de aptitud son:

- Refleje el valor de un individuo de la manera más "real" posible.
- Para dos individuos similares sus respectivos valores en las funciones objetivo sean similares.

Dificultades que podemos encontrar en este paso de los algoritmos genéticos son:

- Existencia de gran cantidad de óptimos locales.
- Que el óptimo global se encuentre muy aislado.
- Si existen gran cantidad de restricciones, gran parte de los puntos del espacio de búsqueda representan individuos no válidos.

En los problemas con restricciones hay que tener en cuenta que algunos individuos representen soluciones no válidas, así que antes de pasar al siguiente punto del algoritmo hay que realizar obligatoriamente:

- Enfoque absolutista: Aquellos individuos que no verifican las restricciones no son considerados como tales y se siguen efectuando cruces y mutaciones hasta obtener individuos válidos, o bien a dichos individuos se les asigna una función objetivo igual a cero. No hay que despreciar a la ligera este enfoque ya que estos individuos pueden contener información claves para poder converger hacia la solución óptima.
- Reconstruir los individuos que no son válidos. Dicha reconstrucción suele llevarse a cabo por medio de un nuevo operador que se acostumbra a denominar reparador. Esto se suele hacer cuando el problema no es un problema de factibilidad (por ejemplo, el TSP).
- Eliminar los individuos que no son válidos.
- Penalización de la función objetivo: Asociarlos un valor de *fitness* $-\infty$ para que tenga una probabilidad menor de cruzarse con otro individuo y con la esperanza que con el paso de las iteraciones del algoritmo ese individuo vaya desapareciendo. Esto se suele hacer cuando el problema es un problema de factibilidad (por ejemplo, el problema de las 8 reinas). También se puede dividir la función objetivo del individuo por una cantidad (la penalización) que guarda relación con las restricciones que dicho individuo viola. Dicha cantidad puede simplemente tener en cuenta el numero de restricciones violadas o bien el denominado costo esperado de reconstrucción, es decir, el coste asociado a la conversión de dicho individuo en otro que no viole ninguna restricción.
- Evaluación aproximada de la función objetivo: En algunos casos la obtención de n funciones objetivo aproximadas puede resultar mejor que la evaluación exacta de una única función objetivo (supuesto el caso de que la evaluación aproximada resulta como mínimo n veces más rápida que la evaluación exacta).

Un problema común en los algoritmos genéticos está relacionado con la velocidad a la que el algoritmo converge hacía una solución óptima. Dependiendo de la velocidad de convergencia, nos encontramos:

- Si la convergencia es muy rápida se produce la convergencia prematura, en la cual el algoritmo genético converge hacia óptimos locales. Una posible solución a estos problemas pasa por efectuar transformaciones en la función objetivo. Por medio de una transformación de la función objetivo, en este caso una comprensión del rango de variación de la función objetivo, se pretende que dichos "superindividuos" no lleguen a dominar a la población.
- El caso contrario consistiría en una convergencia lenta del algoritmo. El problema de la lenta convergencia del algoritmo se resolvería de manera análoga a la convergencia prematura, pero en este caso efectuando una expansión del rango de la función objetivo.

Estos problemas se volverán a tratar en el siguiente apartado de selección.

4.4 Selección

En este punto se trata de generar varios grupos de dos individuos, para luego realizar el cruce entre ellos. A veces se le asocia una probabilidad de emparejamiento, es decir, que a veces el emparejamiento se produce y a veces no. Para ver si se pueden emparejar dos individuos se genera un número aleatorio entre el intervalo de [0, 1] y si el número generado es menor que la probabilidad de emparejamiento, entonces se produce el emparejamiento, sino, no se produce. Debemos garantizar que los mejores individuos tengan una mayor posibilidad de reproducirse frente a los individuos menos aptos. Mientras más apto sea un organismo, más veces será seleccionado para reproducirse. Sin embargo, también hay que dar una probabilidad de seleccionar a los individuos menos aptos ya que éstos pueden incluir información útil en los genes para alcanzar la solución óptima. La presión selectiva afecta al proceso de búsqueda de las siguientes formas:

- Selección muy fuerte: También conocido como peligro de convergencia prematura (*premature convergence*) a un subóptimo debido a que individuos muy aptos dominan la población completa muy rápidamente a medida que avanza la ejecución del algoritmo. Esto ya ha sido comentado en el apartado anterior de *fitness*, con la consecuencia de que disminuye la diversidad necesaria para progresar.
- Selección muy débil: También conocida como baja presión selectiva (*selection pressure*) cuando los valores de aptitud son muy cercanos entre sí, es decir, los organismos de bajo *fitness* dominan la población, generando una evolución muy lenta y produciendo estancamiento en el proceso de búsqueda.

Una posible clasificación de los métodos de selección es la siguiente:

- Métodos de selección dinámicos: En los cuales los probabilidades de selección varían de generación a generación (por ejemplo la selección proporcional de la que hablaremos más adelante).
- Métodos de selección estáticos: En los cuales dichas probabilidades permanecen constantes (por ejemplo, la selección basada en rangos).

También se pueden clasificar de la siguiente forma:

- Preservativo: Si el método de selección asegura que todos los individuos tienen asignada una probabilidad de selección distinta de cero.
- El caso contrario se acostumbra a denominarse extintivo.

Otra posible clasificación que usaremos para explicar los distintos métodos de selección es el siguiente:

- Selección elitista: elección directa de los miembros más aptos de cada generación.
- Selección proporcional: Es el tipo de selección más utilizado, en la cual cada individuo tiene una probabilidad de ser seleccionado como padre que es proporcional al valor de su función objetivo. Los individuos más aptos tienen más probabilidad de ser seleccionados aunque los peores individuos también pueden ser elegidos. Estos tipos de selección suelen ser los que presentan normalmente los problemas de selección fuerte y débil. Denotando por $p_{j,t}^{\text{prop}}$ la probabilidad de que el individuo I_t^j sea seleccionado como padre, se obtiene que:

$$\boxed{p_{j,t}^{\text{prop}} = \frac{g(I_t^j)}{\sum_{j=1}^{\lambda} g(I_t^j)}} \tag{9}$$

  - Selección por rueda de ruleta: también conocido como muestreo universal estocástico, este método (Baker, 1987) está basado en la relación de la aptitud de cada individuo y la media con el resto de sus competidores. Utiliza una ruleta siendo los sectores circulares proporcionales a su valor en la función objetivo. Los individuos son seleccionados a partir de marcadores distribuidos de la misma forma y con un comienzo aleatorio.

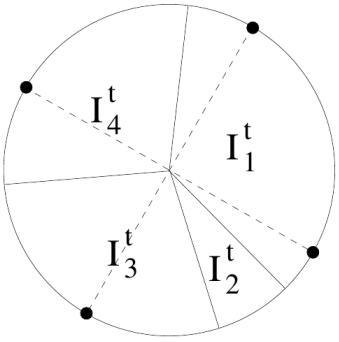


Figura 6: Muestreo universal estocástico

- Selección por rango: mediante varias funciones se le asigna a cada individuo una probabilidad. Este método también entra dentro de la selección proporcional. La probabilidad asignada a cada individuo depende de su rango (*ranking*). Luego se aplica una estrategia de selección proporcional. Este método de selección es una de las maneras de superar el problema relacionado con la rápida convergencia proveniente de los superindividuos, que surge al aplicar la selección proporcional. Aplicando el método por selección por rango se produce una distribución más uniforme de la probabilidad de selección.

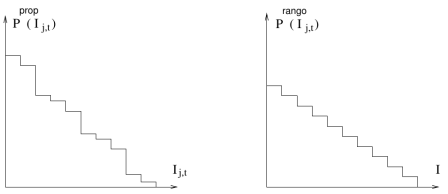


Figura 7: Progresos de selección
Selección de padres proporcional a la aptitud (izquierda) y proporcional al rango (derecha)

- Valor esperado: Este es otro posible refinamiento de la selección proporcional, mediante el cual para cada individuo se introduce un contador inicializado con un valor positivo dependiendo de la media de los valores de aptitud en una generación dada. El contador se reduce una cantidad predeterminada cada vez que la solución es seleccionada para realizar el cruce. En caso de que en una generación dada ese contador llegase a ser negativo, dejará de poder ser escogido para el cruce en esa generación.
- Un esquema introducido por Brindle (1991) denominado muestreo estocástico con reemplazamiento del resto ha proporcionado buenos resultados. En el, cada individuo es seleccionado un número de veces que coincide con la parte entera del número esperado de ocurrencias de dicho suceso, compitiendo los individuos por los restos.

- Selección jerárquica:

  - Selección por torneo: se eligen subgrupo de individuos (con o sin reemplazamiento) y los miembros de cada subgrupo compiten entre ellos. El de mayor valor del subgrupo considerando el *fitness* es el ganador. Solo se elige a un individuo de cada subgrupo para la reproducción y se repite el proceso hasta escoger un número de individuos equivalente al tamaño de la población. Habitualmente el tamaño del torneo es 2, al igual que se permite la selección de individuos de los subgrupos sin que necesariamente sean los mejores.

La selección dirige la búsqueda hacia zonas prometedoras del espacio de búsqueda. Está relacionado con la explotación (presión selectiva). A mayor presión selectiva mayor explotación (menor presión, mayor exploración).

4.5 Cruce

Esta parte está basada en la naturaleza, donde la nueva generación de una especie animal se basa en el intercambio entre los cromosomas de los padres para generar dos hijos. Hay que resaltar que

tanto el cruce como el siguiente paso de mutación y el de *fitness* puede generar o encontrarse hijos que representen soluciones no factibles. Está en nuestra elección corregirlos, eliminarlos, o no hacer nada (aunque esto hace que su función *fitness* sea negativa y a la larga puede ser perjudicial y hacer que el algoritmo nunca encuentre no solo una buena solución, sino que toda la población represente soluciones no válidas). Los operadores clásicos no funcionan de forma óptima con la representación de permutación (la más utilizada en el problema del viajante), sin embargo se han definido tanto operadores de cruce como de mutación válidos para esta representación. Las diversas formas más conocidas de realizar un cruce de cromosomas son:

- Cruce de n puntos: Se seleccionan n puntos y se intercambian los genes que se encuentran entre los puntos. A medida que crece n, crece el número de hijos que se pueden generar de forma exponencial y suele tener un mayor efecto destructivo del cruce (aumenta la probabilidad de ruptura de buenas soluciones) aunque el espacio de búsqueda es propenso a ampliarse. El cruce de 1 punto presenta desventajas como flexibilidad. De Jong (1975) demostró que con este tipo de cruce basado en dos puntos era positivo para el algoritmo genético mientras que el hecho de incrementar el valor de n (puntos) no mejoraba los resultados del algoritmo genético, por lo que en general se ha llegado al consenso de utilizar el cruce de 2 puntos.

$$\text{Padre 1} = 10|1010|1010$$
$$\text{Padre 2} = 11|1000|1110$$

$$\text{Hijo 1} = 11|1010|1110$$
$$\text{Hijo 2} = 10|1000|1010$$

- Cruce uniforme: Propuesto por Syswerda (1991), cada gen del individuo descendiente se crea copiando el gen de uno de los padres, escogido de acuerdo a una "máscara de cruce" generada mediante una muestra aleatoria de tamaño λ, siendo λ el número de elementos del cromosoma extraída de una distribución de probabilidad de Bernoulli de parámetro 0,5. Suponiendo una codificación binaria, cuando aparece un 1 en la "máscara de cruce", el gen del primer padre es copiado, mientras cuando aparezca un 0, se escoge el gen del otro padre. Este tipo de cruce solo genera un hijo.

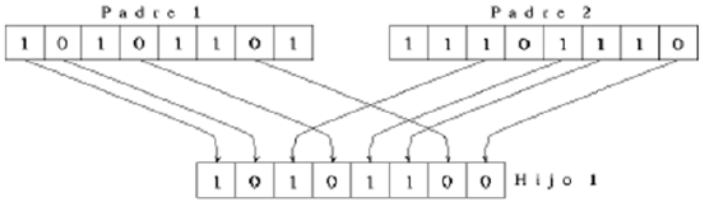


Figura 8: Ejemplo de cruce uniforme

La aplicabilidad de los algoritmos genéticos proviene en gran parte en su capacidad de combinar un proceso de búsqueda global (exploración) con un proceso de búsqueda local (explotación).

- La exploración está relacionada con mayor diversidad de la población, a fin de explorar un amplio espacio de estados no priorizando un alto *fitness*.
- La explotación está relacionada con la focalización en individuos de alto *fitness*, priorizando su importancia.

El cruce recombina las soluciones existentes y está relacionado con la exploración: a mayor número de puntos de cruces implica mayor exploración. A menor número de cruces implica mayor explotación. Los operadores de cruce más utilizados para el problema de viajante son los siguientes, siempre y cuando se utilice una representación basada en una trayectoria, ya que los operadores de cruce y mutación clásicos no son muy efectivos con esta representación. Los tipos de cruce introducidos a continuación se basan en combinar los operadores de cruce clásicos con métodos para intentar guardar el máximo orden posible de subristas de los padres para que no se le tenga que aplicar un operador de reparación y se pueda realizar un cruce satisfactorio:

Representación	Operadores	Autores
Binario	Clásico + operador de reparación	Lidd (1991)
Path	Simple inversion mutation	Holland (1975)
	Partially-mapped crossover	Goldberg and Lingle (1985)
	Sorted match crossover	Brady (1985)
	Order-crossover	Davis (1985)
	Heuristic crossover	Grefenstette (1987b)
	Maximal preservative crossover	Mühlenbein et al. (1988)
	Insertion mutation	Fogel (1988)
	Voting recombination crossover	Mühlenbein (1989)
	Edge recombination crossover	Whitley et al. (1989)
	Inversion mutation	Fogel (1990)
	Exchange mutation	Banzhaf (1990)
	Order based crossover	Syswerda (1991)
	Scramble mutation	Syswerda (1991)
	Position based crossover	Syswerda (1991)
	Displacement mutation	Michalewizc (1992)
	Alternating-positions crossover	Larrañaga et al. (1996a)
Adjacencia	Heuristic crossover 1	Grefenstette et al. (1985)
	Subtour chunks crossover	Grefenstette et al. (1985)
	Alternating edge crossover	Grefenstette et al. (1985)
	Heuristic crossover 3	Suh and Van Gucht (1987)
	Heuristic crossover 2	Jog et al. (1989)
Ordinario	Classical operators	Grefenstette et al. (1985)
Matriz	Repair operators	Seniw (1981)
	Union crossover operator	Fox and McMahon (1987)
	Intersection crossover operator	Fox and McMahon (1987)
	Heuristic inversion mutation	Homaifar and Guan (1991)
	Repair operators	Homaifar and Guan (1991)

Cuadro 2: Resumen de representaciones y operadores (cruce + mutación) (variador)

- *Partially mapped crossover* (PMX): Fue introducido por Goldberg y Lingle (1985). Una parte del *path* que representa a un padre se hace corresponder con una parte de igual tamaño del *path* del otro padre, intercambiándose la información restante (cruce de n puntos). Se genera una correspondencia entre los genes comprendidos entre los puntos de corte. Luego el descendiente i-ésimo se llena copiando los elementos del i-ésimo padre. En el caso de que una ciudad esté ya presente en el descendiente, se reemplaza teniendo en cuenta la correspondencia anterior.

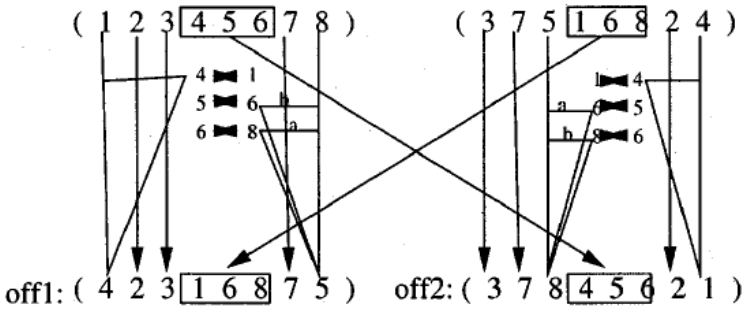


Figura 9: Cruce parcialmente mapeado

- Operador basado en ciclos (CX).
- Operador basado en el orden (OX1).
- Operador basado en el orden (OX2).
- Operador basado en la posición (POS).
- Operador basado en la alternancia de las posiciones (AP).

4.6 Mutación

La mutación consiste en un operador que mediante un valor de probabilidad para modificar los genes de los individuos aporta exploración en la vecindad alrededor suya. El valor óptimo para la probabilidad de mutación es una cuestión que ha generado debate en varios trabajos:

- De Jong (1975) recomienda la utilización de una probabilidad de mutación del bit de l^{-1}, siendo l la longitud del *path*.
- Schaffer y col. (1989) estimaron un valor de $1/\lambda^{0,9318}l^{0,4535}$, donde λ denota el número de individuos en la población.
- Davis (1985) demostró que la mutación no es un concepto que gane importancia a medida que el algoritmo genético progresa, sin embargo, varios autores han demostrado que modificando tanto la probabilidad de cruce como de mutación se obtienen mejores resultados a medida que el algoritmo genético progresa.

Este operador introduce innovación en la búsqueda. Se analizan zonas de las soluciones del problema en una zona local. A mayor mutación implica mayor exploración (búsqueda global). A menor mutación implica mayor explotación.

Se aplica con probabilidad p_{mut} para cada gen de cada cromosoma de los hijos generados en el cruce. La probabilidad de mutar (P_{mut}) es normalmente baja teniendo en cuenta lo comentado anteriormente en este mismo punto, generalmente en el rango [0.001...0.05]. Este punto, al igual que en el de aptitud y el de cruce, puede generar individuos no válidos. Está en nuestra elección corregirlos, eliminarlos, o no hacer nada (aunque esto hace que su función *fitness* sea negativa y a la larga puede ser perjudicial y hacer que el algoritmo nunca encuentre no solo una buena solución, sino que toda la población represente soluciones que no son válidas). Hay varios métodos de mutación:

- Operador de mutación basado en el desplazamiento (DM): El operador (Michalewizc, 1997) comienza seleccionando una subristra al azar. Dicha subristra se extrae de la gira y se inserta en un lugar aleatorio. Esta subrista puede estar formada de 1 a $n-1$ elementos siendo n la longitud del individuo. Por ejemplo, si consideramos la gira representada por (1 2 3 4 5 6 7 8) y suponemos que se selecciona la subrista (3 4 5), después de quitar dicha subrista tenemos (1 2 6 7 8). Supongamos que aleatoriamente seleccionamos la ciudad 7 para insertar a partir de ella la subgira extraída. Esto produciría la gira: (1 2 6 7 3 4 5 8).
- Operador de mutación basado en cambios (EM): El operador (Banzhaf, 1990) selecciona al azar dos ciudades en la gira y las cambia de posición. También es conocido como operador de intercambio recíproco. Por ejemplo, si consideramos la gira representada por (1 2 3 4 5 6 7 8) y suponemos que seleccionamos al azar la tercer y la quinta ciudad. El resultado del operador sobre la gira anterior sería (1 2 5 4 3 6 7 8).
- Operador de mutación basado en la inserción (ISM): El operador (Fogel, 1993; Michalewizc, 1992) escoge aleatoriamente una ciudad en la gira para a continuación extraer dicha ciudad de la gira, e insertarla en un lugar seleccionado al azar. Por ejemplo, si consideramos de nuevo la gira (1 2 3 4 5 6 7 8) y suponiendo que se seleccione la ciudad 4 para colocarla a continuación de la ciudad 7, el resultado sería (1 2 3 5 6 7 4 8).
- Operador de mutación basado en la inversión simple (SIM): El operador (Holland, 1975; Grefenstette y col., 1985) selecciona aleatoriamente dos puntos de corte en la ristra, para a continuación revertir la subristra comprendida entre ambos. Por ejemplo, si consideramos la gira (1 2 3 4 5 6 7 8) y suponemos que el primer punto de corte se escoge entre la segunda y

tercera ciudad, y el segundo punto de corte se escoge entre la quinta y la sexta ciudad, por lo que la gira resultante sería (1 2 5 4 3 6 7 8).

- Mutación binaria clásica: Al aplicar el operador de mutación, cada bit tiene una probabilidad de ser alterado cercana a cero. Por ejemplo, si consideramos de nuevo la ristra (000 001 010 011 100 101) y suponemos que se mutan el primer y segundo bit, obtenemos como resultado la ristra: (110 001 010 011 100 101) la cual hay que tener en cuenta que no representa una gira.

4.7 Reemplazo de la población

Establece el criterio de los supervivientes en cada iteración/sustitución generacional. Dependiendo del método de selección y cruce, puede que al final de cada iteración la población haya aumentado. Esto debe ser evitado para que no se dispare el número de individuos en esta y aumente de forma innecesaria el tiempo de cómputo del algoritmo. Para ello, una vez obtenidas las soluciones descendientes, la reducción de la población a un determinado tamaño consiste en escoger varios individuos entre la población y los descendientes que estos han generado.

- Generacional: Toda la población es reemplazada en cada generación. Cada individuo sobrevive solo una generación y todos los hijos reemplazan a todos los padres. Este enfoque también es conocido como reducción simple y suele ser el más utilizado junto con el elitista, que consistirá en un elemento clave en los algoritmos genéticos multiobjetivo.
- Estado-estacionario (*Steady-state* AG): Suele ser un reemplazo típico. Solo unos pocos individuos son reemplazados en cada generación. Típicamente se generan uno o dos hijos y solo uno o dos individuos son reemplazados.
- *Gap* generacional: Se trata de un criterio combinado del reemplazo generacional y del reempazo estado-estacionario. Se define un porcentaje (*gap*) de las soluciones que serán reemplazados cada generación, por evaluación del *fitness*.
- Reemplazo catastrofista. Cada cierto tiempo se elimina una gran cantidad de individuos:
  - Empaquetado: Solo se mantiene un individuo de los que tienen igual *fitness*.
  - Juicio final: Se destruye toda la población excepto el más avanzado. El resto de población se genera aleatoriamente.
- Elitismo: Bien se escogen de entre los de la población más sus descendientes, y se escogen los individuos hasta alcanzar el tamaño acordado de la población, que sería λ. Este último enfoque también se le conoce como reducción elitista de grado λ.
- Podemos también considerar otros procedimientos que combinan la reducción simple y elitismo, por ejemplo, si escogemos un número de individuos entre los padres y los descendientes que sea menor que el tamaño de la población y el resto los escogemos de entre los descendientes no seleccionados hasta el momento hasta llegar al tamaño de la población.

El concepto de reducción está ligado con el de la tasa de reemplazamiento generacional, t_{rg}, es decir en el porcentaje de hijos generados con respecto del tamaño de la población.

4.8 Condición de parada

Hay varios factores/condiciones para parar el algoritmo genético como:

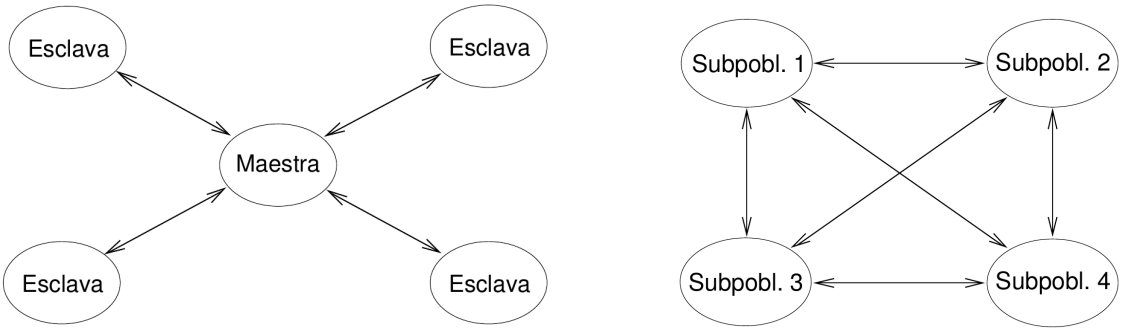
- Número fijo de generaciones o soluciones generadas.
- Tras un tiempo de cómputo fijo.
- Considerando medidas de diversidad en la población (convergencia).
- Cuando se alcance el óptimo (si es conocido), o un determinado valor de *fitness*.
- Tras varias generaciones sin ninguna mejora.

Al parar el algoritmo, se proporciona como solución el individuo con mejor valor de aptitud o *fitness* de la nueva población generada.

4.9 Paralelismo

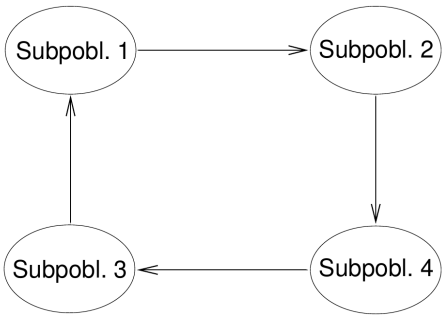
En este punto se introducirán tres modos distintos de aprovechar el paralelismo de los algoritmos mediante los denominados modelos de islas. Cuando veamos los algoritmos genéticos multiobjetivo comprobaremos que la mayoría de ellos se pueden paralelizar de forma muy similar. A pesar del *a priori* poco coste computacional de los algoritmos genéticos, a veces conviene trabajar con poblaciones muy numerosas y el componente que suele requerir más gasto computacional el el cálculo del *fitness* en los individuos. Es por este motivo el cual en ocasiones puede interesar paralelizar la ejecución de los algoritmos genéticos. Los interesados pueden consultar Stender (1993).

El método de paralelización más utilizado en los algoritmos genéticos se basa en los modelos de islas, que consiste en dividir la población en varios grupos (isla), donde en cada una de ellas se puede ejecutar un algoritmo genético distinto. Cada cierto número de generaciones se efectúa un intercambio de información entre los grupos (emigración). La emigración hace que los modelos de islas sean capaces de aprovechar la diversidad entre los grupos, obteniéndose de esta manera diversidad genética. Además se tiene que definir un procedimiento mediante el cual se mueve el material genético de una "isla" a otra. La determinación de la tasa de migración es una cuestión a tener muy en cuenta, ya que de ella puede depender la convergencia prematura de la búsqueda.



(a) Comunicación en estrella

(b) Comunicación en red



(c) Comunicación en anillo

Figura 10: Esquemas de paralelización monobjetivo

En función de la comunicación entre las subpoblaciones se pueden distinguir distintos modelos de islas. Algunas comunicaciones típicas son:

- Comunicación en estrella: Existe una subpoblación que es seleccionada como maestra (aquella que tiene mejor media en el valor de la función objetivo), siendo las demás consideradas como esclavas. Todas las subpoblaciones esclavas mandan sus h_1 mejores individuos ($h_1 \geq 1$) a la subpoblación maestra la cual a su vez manda sus h_2 mejores individuos ($h_2 \geq 1$) a cada una de las subpoblaciones esclavas.
- Comunicación en red: No existe una jerarquía entre las subpoblaciones, mandando todas y cada una de ellas sus $h_3 (h_3 \geq 1)$ mejores individuos al resto de las subpoblaciones.
- Comunicación en anillo: Cada subpoblación envía sus mejores individuos (siendo este valor siempre mayor o igual a 1) a una población vecina, efectuándose la migración en un único sentido de flujo.

5. Algoritmos genéticos multiobjetivo

Dentro del conjunto de los EMO (*Evolutionary Multiobjective Optimization*) o MOEA (*Multiobjective Evolutionary Algorithms*) se encuentran estos distintos tipos de resolver problemas multiobjetivos mediante la evolución. Para una mayor explicación de los mismos, mirar [DEB01].

- Algoritmos genéticos, descritos por primera vez por John Holland.
- Estrategias evolutivas, descritos por primera vez por I. Rechenberg y H.-P. Schwefel. Se basan en las estrategias de los algoritmos genéticos $(1+1)$-ES, (μ, λ)-ES y $(\mu/\mu_I, \lambda)$-ES.
- Programación evolutiva, descritos por primera vez por David Fogel en 1988.
- Programación genética, descritos por primera vez por Koza.

En los algoritmos evolutivos monobjetivo, la función objetivo y la función de aptitud son iguales, mientras que en los algoritmos genéticos multiobjetivo la asignación del *fitness* y la selección deben cubrir varias funciones objetivo diferentes, por lo tanto estas dos técnicas variarán respecto a los algoritmos genéticos monobjetivo. En definitiva, consiste en encontrar un vector de variables de decisión que satisfagan restricciones y optimicen una función vectorial cuyos elementos representan funciones objetivo. Al tratarse de un problema de optimización multiobjetivo, ya no solo se busca una única solución óptima, sino un conjunto de soluciones posibles de calidad equivalente. El espacio de búsqueda se convierte en parcialmente ordenado, donde existe un conjunto de soluciones compensadas (*trade-offs*) entre los objetivos conflictivos, que puedan ser aceptadas por el tomador de decisiones. Es muy inusual que un punto optimice simultáneamente todos los objetivos de un problema multiobjetivo. La noción más adoptada es la descrita por Francis Ysidro Edgeworth (1881) que luego fue generalizada por Pareto (1896) y es conocida como Pareto óptimo.

Los diversos motivos que justifican el usar algoritmos evolutivos en problemas multiobjetivo son los siguientes:

- Por su característica de búsqueda paralela (poblacional) que abre la opción de encontrar varias soluciones del conjunto óptimo Pareto en una ejecución.
- Pueden tratar problemas de forma o discontinuidad en la frontera de Pareto que no son posibles al emplear técnicas de programación matemática clásicas (Coello, 1999).
- La gran mayoría de ellos se pueden paralelizar.

A parte de los conceptos que se van a comentar a continuación, hay que indicar que hay más aspectos opcionales a tener en cuenta para la aplicación de algoritmos genéticos multiobjetivo como los tipos de selección (selección de entorno/selección de apareamiento) además del manejo de constantes y articulación de preferencia.

5.1 Óptimo de Pareto

Antes de analizar más detenidamente los problemas multiobjetivo, hay que dejar claros los siguientes conceptos relacionados con la dominancia: Pareto dominancia, Pareto óptimo y frente de Pareto.

Se dice que una solución a domina a otro b (se denota como $a \preceq b$) si y solo si (suponiendo maximización):

$$\boxed{\forall i \in 1, 2, ..., K \mid f_i(a) \geq f_i(b) \wedge \exists j \in 1, 2, ..., K \mid f_j(a) > f_j(b)} \tag{10}$$

Es decir, una solución se considera Pareto óptima o no dominada si es mejor o igual en todos los objetivos y al menos mejor en uno de ellos. Si ninguno de los vectores domina al otro se considera que los dos individuos son incomparables (ni uno ni el otro es una solución Pareto óptima).

Una solución $x \in X_f$ es Pareto optimal si no es dominada por ninguna otra solución del espacio (el conjunto Ω).

$$\boxed{\nexists x' \in \Omega \text{ para el cual } v' = F(x') \text{ domina a } u = F(x)} \tag{11}$$

donde u es un vector objetivo $u = (u_1, ..., u_k)$.

El conjunto de todas las soluciones no dominadas $X^* \subset \boldsymbol{X}$ se le conoce como el conjunto Pareto optimal. Dado un problema de optimización multiobjetivo $F(x)$, el conjunto Pareto óptimo, denotado por $\mathcal{P}^*$ se define como:

$$\boxed{\mathcal{P}^* = \{x \in X_f |\ \nexists x' \in X_f \text{ para el cual } F(x') \text{ domine a } F(x)\}} \tag{12}$$

El frente Pareto óptimo está compuesto por todos los puntos del conjunto Pareto óptimo y es conocido como $\mathcal{PF}^*$. Los vectores de valores de las funciones objetivo de los elementos del conjunto Pareto optimal $z(X^*) \subset \boldsymbol{Y}$ forman la frontera (o el frente) de Pareto, que conforman el espacio de soluciones y forman parte del conjunto del espacio objetivo. Las claves para definir correctamente mediante individuos un frente de Pareto óptimo son:

- Debe estar tan cerca del frente de Pareto óptimo como sea posible.
- Las soluciones deben estar uniformemente distribuidas sobre el frente. Esto es lo que aporta diversidad a la población del algoritmo genético, un concepto importante en los algoritmos genéticos multiobjetivo. Esto permitirá presentar un conjunto de soluciones con diferentes compromisos entre los criterios que se tienen que optimizar. No es considerado como una buena solución del algoritmo aquel que proporciona muy buenas soluciones (muy cercanas al frente de Pareto) pero todas casi idénticas.
- La aproximación debe capturar todo el frente de Pareto, incluyendo los extremos.

Un ejemplo de problema MOP para encontrar el óptimo de Pareto es el siguiente. Un productor desea minimizar costos de producción f_1 y distribución f_2 obteniendo las siguientes soluciones:

- A: $f_1 = 2, f_2 = 10$
- B: $f_1 = 4, f_2 = 6$
- C: $f_1 = 8, f_2 = 4$
- D: $f_1 = 9, f_2 = 5$
- E: $f_1 = 7, f_2 = 8$

Se observa que A, B y C son soluciones no dominadas, mientras que D y E son dominadas por C y B: $C \prec D, B \prec E$. El algoritmo más eficiente para encontrar las soluciones no dominadas es el método de bisección recursiva (algoritmo de Kung [MEN07]). Hay que resaltar que la frontera puede delimitar una área, ser una frontera formada por una línea con dos extremos (el caso más habitual), etc. es decir, cualquier cuerpo geométrico existente en la representación multidimensional donde se represente el espacio objetivo.

Antes de continuar hay que dejar claros las siguientes denominaciones:

- Al conjunto Pareto óptimo de una generación se le denomina $P_{known}(t)$, donde t representa el número de generaciones transcurridas desde el inicio del algoritmo genético multiobjetivo.
- El frente de Pareto correspondiente a $P_{known}(t)$ se denota como $PF_{known}(t)$.
- El conjunto Pareto obtenido al final de la ejecución del algoritmo genético multiobjetivo se denota como P_{known}.
- El frente de Pareto obtenido al final de la ejecución del algoritmo genético multiobjetivo se denota como PF_{known}.

Cuando se resuelve un algoritmo genético multiobjetivo se debe cumplir mínimo una de las siguientes relaciones: $PF_{known} = \mathcal{PF}^*, PF_{known} \subset \mathcal{PF}^*$ o $PF_{known} \approx \mathcal{PF}^*$, donde $\mathcal{PF}^*$ representa el frente de Pareto óptimo real.

Buscar el frente de Pareto en un espacio objetivo $\boldsymbol{Y}$ normalmente no es práctico dependiendo del tamaño, por lo que a veces se necesita ser menos ambicioso. El motivo teórico para usar la aproximación del conjunto ε-Pareto será explicado en la siguiente sección de convergencia de los

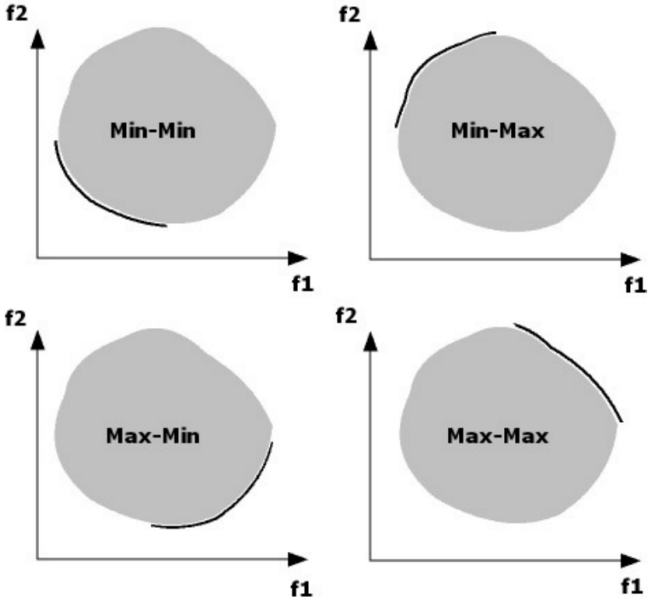


Figura 11: Tipologías de frentes óptimos de Pareto

algoritmos genéticos multiobjetivo. La aproximación del conjunto ε-Pareto fue propuesta como concepto de solución práctica y no solo representa todos los vectores $\boldsymbol{Y}^*$ pero también contiene un pequeño número de elementos. El frente ε-Pareto está basado en la siguiente generalización de la relación de dominancia.

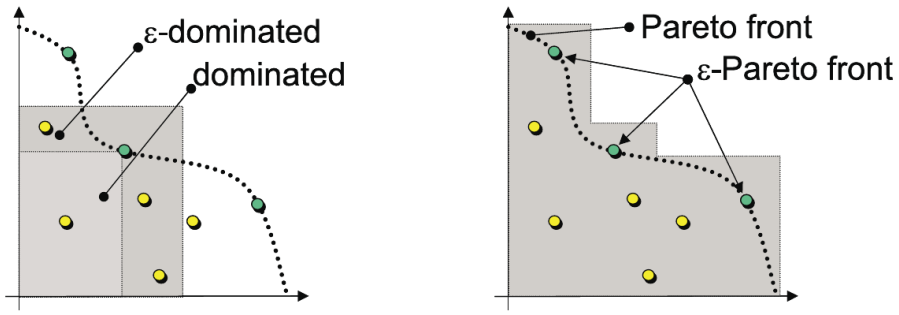


Figura 12: Concepto de ε-dominancia y frente ε-Pareto

La ε-dominancia, teniendo $\boldsymbol{a}, \boldsymbol{b} \in \boldsymbol{Y}$, entonces $\boldsymbol{a}$ se dice que ε-domina $\boldsymbol{b}$ para algún $\varepsilon > 0$, denotado como $\boldsymbol{a} \succ_\varepsilon \boldsymbol{b}$ si

$$\boxed{\varepsilon \cdot a_i \geq b_i \quad \forall i \in \{1, ..., k\}} \tag{13}$$

Si $\boldsymbol{Y} \subseteq \mathbb{R}^{+^k}$ es un conjunto de vectores y $\varepsilon \geq 1$. Entonces el conjunto $\boldsymbol{Y}_\varepsilon$ es llamado una aproximación frente ε-Pareto de $\boldsymbol{Y}$, si cualquier vector $\boldsymbol{b} \in \boldsymbol{Y}$ es ε-dominado por mínimo un vector $\boldsymbol{a} \in \boldsymbol{Y}_\varepsilon$

$$\boxed{\forall \boldsymbol{b} \in \boldsymbol{Y} \ \exists \boldsymbol{a} \in \boldsymbol{Y}_\varepsilon : \quad \boldsymbol{a} \succ_\varepsilon \boldsymbol{b}} \tag{14}$$

El conjunto de todas las aproximaciones de frentes ε-Pareto de $\boldsymbol{Y}$ es denotado como $\boldsymbol{P}_\varepsilon(\boldsymbol{Y})$.

Por supuesto, el conjunto $\boldsymbol{Y}_\varepsilon$ no es único. Diferentes conceptos de la ε-eficiencia y la correspondiente aproximación del frente de Pareto existen en la literatura. La mayoría de conceptos tratan con conjuntos infinitos, y no son prácticos para nuestra proposición de producir y mantener un subconjunto representativo. Sin embargo, es de interés teórico y tiene buenas propiedades que pueden en instancia ser usadas en pruebas de convergencia.

Hay que tener en cuenta que este concepto de aproximación puede ser usado con definiciones

de ε dominancia con pocas diferencias, por ejemplo la siguiente aproximación aditiva

$$\boxed{\varepsilon + a_i \geq b_i \quad \forall i \in \{1, ..., k\}} \tag{15}$$

donde ε_i son constantes, definidas separadamente para cada coordenada.

Un refinamiento del concepto de ε aproximación de conjuntos de Pareto conduce a la siguiente definición.

Si $\boldsymbol{Y} \subseteq \mathbb{R}^{+^m}$ es un conjunto de vectores y $\varepsilon > 0$. Entonces el conjunto $\boldsymbol{Y}^*_\varepsilon \subseteq \boldsymbol{Y}$ es llamado un frente ε-Pareto de $\boldsymbol{Y}$ si

1. $\boldsymbol{Y}^*_\varepsilon$ es una ε aproximación del conjunto de Pareto de $\boldsymbol{Y}$, por ejemplo, $\boldsymbol{Y}^*_\varepsilon \in \boldsymbol{P}_\varepsilon(\boldsymbol{Y})$, y
2. $\boldsymbol{Y}^*_\varepsilon$ solo contiene Pareto puntos de $\boldsymbol{Y}$, por ejemplo, $\boldsymbol{Y}^*_\varepsilon \subseteq \boldsymbol{Y}^*$.

El conjunto de todos los frentes ε-Pareto de $\boldsymbol{Y}$ es denotado como $\boldsymbol{P}^*_\varepsilon(\boldsymbol{Y})$.

5.2 Convergencia

El límite dentro de los MOEAs es el interés cuando queremos investigar la propiedades de su convergencia global. Se refiere a la cuestión de los límites que el algoritmo puede conseguir cuando los recursos disponibles son ilimitados.

Robustamente hablando, un MOEA es denominado globalmente convergente si la secuencia del frente de Pareto $\mathbf{A}^{(t)}$ produce la convergencia al verdadero frente de Pareto $\boldsymbol{Y}^*$ mientras el número de generaciones t sea infinito. Esto es intuitivamente claro que esta propiedad solo puede ser cumplida con memoria ilimitada como la cantidad de frentes de Pareto puede ser numeroso en general. Implementaciones prácticas, sin embargo, siempre tienen que tratar con memoria limitada. En este caso es restringida a buscar un subconjunto del frente de Pareto, y la convergencia global del algoritmo debe garantizar $\mathbf{A}^{(t)} \longrightarrow \mathbf{Y}^t \subseteq \boldsymbol{Y}^*$.

En el caso monobjetivo, dos condiciones son suficientes para garantizar la convergencia global:

1. Una estricta distribución de mutación, que asegure que cualquier solución $\mathbf{x}' \in \boldsymbol{X}$ pueda ser producida para cada $\mathbf{x} \in \boldsymbol{X}$ por mutación con una probabilidad positiva.
2. Un entorno elitista para el operador de selección, que asegure que una solución óptima no se pierde y que no se produzca deteriorización.

Mientras que la condición de mutación se extrapola fácilmente al caso multiobjetivo, el operador de selección elitista no lo hace. Esto se debe al hecho de que el orden total de soluciones no es dado nunca y las soluciones pueden convertirse incomparables con el resto. Si hay muchas soluciones no dominadas que surgen que pueden ser guardadas en la población, algunas deben ser descartadas. Este entorno de estrategia de selección esencialmente determina si un algoritmo converge globalmente o no.

Rudolph, Hanne y Rudolph & Agapie han propuesto diferentes esquemas de selección que imposibilitan la deteriorización y garantizan la convergencia. La idea básica es que esas soluciones sean descartadas solo si hay un reemplazo alternativo. Esto asegura la suficiente monotonía en la secuencia de soluciones aceptadas.

La mayoría de estados del arte de los MOEAs tienen en cuenta la información de densidad en el criterio de dominancia. Sin embargo, para la mayoría de estos algoritmos está probado que los pasos de selección pueden llevar a la deteriorización. Por lo tanto, la convergencia no puede ser garantizada para cada uno de esos algoritmos.

Basado en el concepto del frente de Pareto, una estrategia de selección puede ser construida para que cumpla la segunda condición de la convergencia global. El siguiente algoritmo tiene dos niveles conceptuales. En el nivel grueso, el espacio de búsqueda es discretizado por una división en cajas, donde cada vector únicamente pertenece a una caja. Usando una relación generalizada de dominancia en estas cajas, el algoritmo siempre mantiene un conjunto de cajas no dominadas, que

garantiza la propiedad de la ε-aproximación. En el mejor nivel a lo sumo un elemento es mantenido en cada caja. Dentro de la caja, cada vector representativo puede solo reemplazado por otro dominante, garantizando la convergencia.

Data: $\boldsymbol{A}$, $\boldsymbol{y}$
Result: $\boldsymbol{A}$'
D := $\{\boldsymbol{y'} \in \boldsymbol{A} \mid box(\boldsymbol{y}) \succ box(\boldsymbol{y'})\}$;
if $D \neq \emptyset$ **then**
  $\boldsymbol{A'} := \boldsymbol{A} \cup \{\boldsymbol{y}\} \setminus$ D;
else if $\exists \boldsymbol{y'} : (box(\boldsymbol{y'}) = box(\boldsymbol{y}) \wedge \boldsymbol{y} \succ \boldsymbol{y'})$ **then**
  $\boldsymbol{A'} := \boldsymbol{A} \cup \{\boldsymbol{y}\} \setminus \{\boldsymbol{y'}\}$;
else if $\nexists \boldsymbol{y'} : box(\boldsymbol{y'}) = box(\boldsymbol{y}) \vee box(\boldsymbol{y'}) \succ box(\boldsymbol{y})$ **then**
  $\boldsymbol{A'} := \boldsymbol{A} \cup \{\boldsymbol{y}\}$;
else
  $\boldsymbol{A'} := \boldsymbol{A}$;
end

Algoritmo 2: Función de selección del frente ε-Pareto

Data: $\boldsymbol{y}$
Result: b {box index vector}
for $i \in \{1, ..., k\}$ **do**
  $b_1 := \lfloor \frac{\log \boldsymbol{y}_i}{\log \varepsilon} \rfloor$;
end
$b := (b_1, ..., b_k)$;

Algoritmo 3: Función de cajas

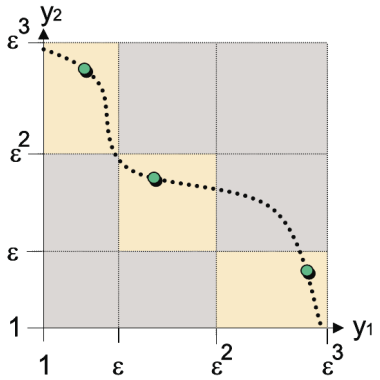


Figura 13: Grid en el espacio objetivo inducido por los algoritmos

El siguiente teorema muestra que un MOEA usando la estrategia de selección comentada anteriormente cumple con el criterio de monotonicidad y nunca pierde soluciones 'importantes'.

Teniendo $\mathbf{Y}^{(t)} = \bigcup_{j=1}^{t} \mathbf{y}^{(j)}, 1 \leq \mathbf{y}_i^{(j)} \leq B$ como el conjunto de todos los vectores objetivo creados por un MOEA y dado el operador de selección definido en el siguiente algoritmo, entonces $\mathbf{A}^{(t)}$ es un conjunto ε-Pareto de $\mathbf{Y}^{(t)}$ con tamaño fijo,

1. $\mathbf{A}^{(t)} \in \mathbf{P}_{\varepsilon}^{*}(\mathbf{Y}^{(t)})$
2. $|\mathbf{A}^{(t)}| \leq (\frac{\log B}{\log \varepsilon})^{(k-1)}$

Ahora, si la distribución de mutación garantiza que cada solución será producida, podemos probar la convergencia global de ese MOEA. Usando la estrategia de archivo manteniendo un frente ε-Pareto, el usuario puede estar seguro de tener una representativa y bien distribuida aproximación, además de un verdadero algoritmo elitista en el sentido que ninguna mejor solución ha sido encontrada y subsecuencialmente perdida durante la ejecución.

5.3 Preservación de la diversidad

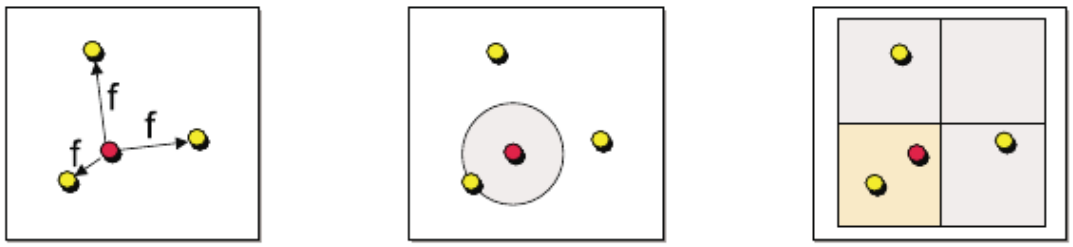


Figura 14: Kernel/vecino más cercano/hiperred

La preservación de soluciones lo más diversas posibles es un objetivo en el diseño de MOEA. La mayoría de los algoritmos genéticos multiobjetivo intentan mantener la diversidad dentro del conjunto de Pareto actual considerando información de densidad en el proceso de selección: la probabilidad de que un individuo sea seleccionado disminuye cuanto mayor es la densidad de individuos en su vecindario. Este problema está estrechamente relacionado con la estimación de la probabilidad de las funciones de densidad en las estadísticas, y los métodos utilizados en los algoritmos genéticos multiobjetivo se pueden clasificar de acuerdo con las categorías para las técnicas de estimación de la densidad estadística.

Algunos de ellos trabajan con información acerca de la densidad de las soluciones donde la posibilidad de que una solución sea seleccionada es inversamente proporcional a la densidad de individuos en la zona. De algún modo algunos MOEA pueden ser clasificados como técnicas de estimación estadística de la densidad.

Los métodos Kernel definen la vecindad de un punto en términos de la llamada función Kernel K. Para cada individuo se calculan las distancias d_i a todos los demás individuos y después de aplicar K se suman los valores resultantes $K(d_i)$. La suma de los valores de la función K representa la estimación de la densidad para el individuo. El uso compartido de la aptitud es la técnica más popular de este tipo en el campo de la computación revolucionaria, que se basa, por ejemplo, en MOGA, NSGA y NPGA.

Las técnicas del vecino más cercano toman en cuenta la distancia de un punto dado a su vecino k más cercano para estimar la densidad de individuos de la solución. Por lo general, el estimador es una función de la inversa de esta distancia. SPEA2, por ejemplo, calcula para cada individuo la distancia al k-ésimo individuo más cercano y agrega el valor recíproco al valor bruto de aptitud física (se debe minimizar la aptitud física).

Los histogramas definen una tercera categoría de estimadores de densidad del entorno de la solución que usan una hiperred para definir vecindarios dentro del espacio. La densidad alrededor de un individuo se calcula simplemente por el número de individuos en la misma caja de la cuadrícula, es decir, se hace una división cartesiana y se cuentan cuantos puntos hay en cada división. La hiperred puede ser fija, aunque generalmente está adaptada con respecto a la población actual como, por ejemplo, en PAES.

Sin embargo, debido a limitaciones de espacio, no se puede proporcionar aquí una discusión sobre las ventajas e inconvenientes de los diversos métodos (el lector interesado es referido al libro de Silverman). Además, tenga en cuenta que todos los métodos anteriores requieren una medida de distancia que se puede definir en el genotipo, en el fenotipo con respecto al espacio de decisión, o en el fenotipo con respecto al espacio objetivo.

La mayoría de los enfoques consideran la distancia entre dos individuos como la distancia entre los vectores objetivos correspondientes. Otras técnicas utilizadas en la preservación de la diversidad son las siguientes (la mayoría basadas en inducir a la formación de nichos):

- *Fitness sharing*: Esquematizado por Goldberg y Richardson. (1987) y Deb (1989), se puede

implementar tanto en el espacio de decisión como en el espacio de objetivos. NPGA y NSGA utilizan está técnica. Este procedimiento está inspirado en las ideas de Holland sobre la formación de especies y nichos dentro de ciertos ecosistemas mediante la distribución de recursos disponibles, donde se considera que el *fitness* es un recurso general que pertenece a todos los individuos de un nicho y por tanto puede compartirse entre ellos.

Un nicho está conformado por individuos genéticamente parecidos o de "la misma especie". En el proceso de *fitness sharing* cada individuo en un nicho comparte su aptitud o *fitness* con el resto de individuos del mismo nicho, de tal modo los nichos con mayores valores de *fitness* son capaces de mantener un mayor número de individuos. Al compartirse el *fitness* entre todos los individuos del nicho, éstos tienen en particular un menor valor de *fitness* del que tendría si no se utilizase el *fitness sharing*. De este modo se mantiene el equilibrio natural y un nicho no tiende a mantener una cantidad de individuos superior a la que puede soportar.

Además de requerir algún modo de medir la similitud genética (distancia) entre los individuos para indicar qué individuos pertenecen a qué nichos, también se necesita conocer una distancia conocida como radio del nicho (σ_{share}), es decir, una distancia umbral para definir los límites de cada nicho.

Para realizar su cálculo, el valor de *fitness* real de un individuo $i \in \mathbf{X}(F'(i))$ se computa a partir de su *fitness* original ($F(i)$), dividido entre su contador de nicho cn_i. El contador de nicho estima la cantidad de individuos que pertenecen a un mismo nicho. Este valor se obtiene sumando valores indicados por una "*sharing function*" que recibe como parámetros al individuo en cuestión y al resto de los individuos de la población. Si dos elementos de la población son idénticos, la "*sharing function*" devuelve 1. Si dos elementos de la población exceden el radio del nicho, la "*sharing function*" devuelve 0, indicando que los individuos pertenecen a nichos diferentes y por tanto no comparten sus valores de *fitness*. En el otro caso, la "*sharing function*" devuelve un valor intermedio entre 0 y 1. Una "*sharing function*" comúnmente utilizada es la siguiente

$$\boxed{\begin{array}{ll} Sh(d_{ij}) = 1 - \left(\dfrac{d_{ij}}{\sigma_{share}}\right)^{\alpha} & \text{si } d_{ij} \leq \sigma_{share} \quad \text{o} \\ Sh(d_{ij}) = 0 & \text{en otro caso} \end{array}} \tag{16}$$

El valor típico para la potencia de la división en la mayoría de los casos es $\alpha = 2$. A continuación, el *fitness* degradado de un individuo i está dado por:

$$\boxed{F'(i) = \frac{F(i)}{cn_i}} \tag{17}$$

donde cn_i es el número de nicho del individuo $i \in \mathbf{X}$ que se calcula de la siguiente manera:

$$\boxed{cn_i = \sum_{j=1}^{|X|} Sh(d_{ij})} \tag{18}$$

y la distancia d_{ij} es la distancia euclidiana o la distancia de Hamming entre las soluciones i y $j \in \mathbf{X}$ (dependiendo del problema) casi siempre realizadas en el espacio objetivo.

Para asignar el valor de aptitud real a cada solución de la población, se realiza iterativamente para cada rango, del mejor al peor:

- A los individuos del rango analizado se le asigna el *fitness* original del rango anterior excepto si estamos analizando el primer rango, al cual se le asigna como *fitness* original el número de soluciones de la población.
- Se calcula el valor mínimo de aptitud real asignados a los individuos del rango y a este valor se le resta un pequeño valor positivo ϵ. Para el siguiente mejor rango empezarán asignándose como *fitness* original este valor calculado.

Una de las principales desventajas que presenta en los algoritmos genéticos multiobjetivo es el decidir el parámetro σ_{share} o radio del nicho. Una variante del *fitness sharing* es el "*niching* secuencial" propuesta por Beasley, Bull y Martin. El "*fitness sharing*" se dejó a un lado para probar otras técnicas más eficaces como el *crowding*.

- *Crowding*: Introducido por De Jong, este método consiste en:
  - Escoger una cantidad de elementos mediante selección proporcional al valor de aptitud, por lo que se generarán un número igual de hijos.
  - Por cada hijo se toma una muestra de la población actual de tamaño indicado por un factor de *crowding*.
  - Por último la descendencia reemplaza a los individuos de la muestra escogida anteriormente con los que tienen mayor similitud o menos distancia.

  El método para calcular la distancia escogido por De Jong era la distancia de Hamming con valores típicos del factor de *crowding* de 2 o 3, o incluso 10 % del tamaño de la población actual, sin embargo con este método se ha probado que no suele presentar buenos resultados. Mahfoud ha propuesto una variación conocida como *crowding* determinístico. En este nuevo método:
  - Se generan pares aleatoriamente considerando a toda la población.
  - Se genera la descendencia aplicando los operadores de cruce y mutación.
  - Cada hijo compite con su padre mediante dos torneos basados en la distancia entre ellos y el ganador pasa a la siguiente generación.

  Este método tiene capacidad de mantener la diversidad en la población y agrupar los individuos de la misma en nichos, aunque también se ha demostrado que tiene limitaciones.
- Restricciones para el emparejamiento.
- Aislamiento por distancia.
- Formación de subespecies dentro de las especies (*overspecification*).
- Reinicialización.

5.4 Asignamiento de fitness

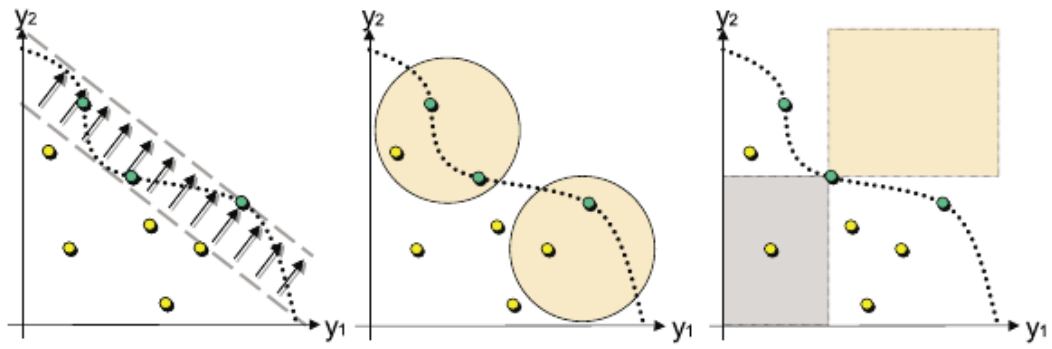


Figura 15: Estrategias de asignamiento de fitness
La primera imagen consiste en la basada en la agregación como la suma de pesos, la segunda es la basada en criterio como VEGA y la tercera la basada en dominancia como SPEA2

En contraste con la optimización de un solo objetivo donde la función objetivo y la función de *fitness* son en la mayoría de casos iguales, tanto la asignación como la selección de aptitud deben permitir varios objetivos con problemas de optimización multicriterio. El paso de asignación de *fitness* y el de selección son los únicos que varían respecto a los algoritmos genéticos monobjetivo.

Hay que tener en bastante consideración este paso, ya que tanto en los algoritmos genéticos monobjetivo como en los multiobjetivo (depende del algoritmo), la mayor parte de la carga computacional reside en el cálculo del *fitness* de los individuos. En general, se pueden distinguir las asignaciones de aptitud basadas en la agregación, basadas en criterios y basadas en Pareto.

Un enfoque que se basa en las técnicas tradicionales para generar superficies *trade-off* es agregar los objetivos en una sola función objetivo que combina todas las funciones objetivo. Los parámetros de esta función se modifican sistemáticamente durante la ejecución de optimización para encontrar un conjunto de soluciones no dominadas en vez de una única solución de compromiso. Por ejemplo, algunos algoritmos genéticos multiobjetivo usan agregación de suma ponderada, donde los pesos representan los parámetros que se cambian durante el proceso de evolución.

Los métodos basados en criterios cambian entre los objetivos durante la fase de selección. Cada vez que se elige un individuo para la reproducción, un objetivo potencialmente diferente decidirá qué miembro de la población se copiará en el grupo de apareamiento. Por ejemplo, Schaffer propuso llenar porciones iguales del grupo de apareamiento de acuerdo a los distintos objetivos, mientras que Kursawe sugirió asignar una probabilidad a cada objetivo que determina si el objetivo será el criterio de clasificación en el siguiente paso de selección: el las probabilidades pueden ser definidas por el usuario o elegidas al azar a lo largo del tiempo.

La idea de calcular la aptitud de un individuo sobre la base del dominio de Pareto se remonta a Goldberg y se han propuesto diversas formas de explotar el orden parcial en la población. Algunos enfoques usan el rango de dominancia, es decir, el número de individuos por los cuales un individuo está dominado, para determinar los valores de aptitud. Otros hacen uso de profundidad de dominación; aquí, la población se divide en varios frentes y la profundidad refleja a qué frente pertenece un individuo. Alternativamente, también se puede tener en cuenta el recuento de dominancia, es decir, el número de individuos dominados por un individuo determinado. Por ejemplo, SPEA y SPEA2 asignan valores de aptitud física en función del rango de dominancia y el recuento. Independiente de la técnica utilizada, la aptitud física está relacionada con toda la población, en contraste con los métodos basados en la agregación que calculan el valor de la aptitud física de un individuo independientemente de otros individuos.

5.4.1 Basados en agregación

Todas las funciones objetivo se combinan en una única función objetivo. Los pesos de esta función son modificados durante la ejecución del algoritmo genético para obtener varias soluciones no dominadas en vez de una sola. Este tipo de métodos tienen como desventaja que no suelen encontrar el número necesario de soluciones para representar de forma correcta el frente de Pareto.

5.4.2 Basados en criterio

Cuando un individuo es seleccionado para generar descendencia, una función objetivo diferente en cada iteración decidirá si ese individuo pertenecerá o no a la siguiente generación. Un ejemplo de algoritmo genético que entraría en esta categoría sería VEGA.

5.4.3 Basados en Pareto

Estos tipos de algoritmos se puede basar en dos estrategias:

- Unos hacen uso de la profundidad de dominancia, es decir, la población es dividida en varios frentes y la profundidad indica a que frente pertenece el individuo. En todas las técnicas de este tipo el *fitness* está relacionado con la población. Un ejemplo de algoritmo genético que entraría en esta categoría sería PAES.
- Otros utilizan el *ranking* de dominancia, es decir, el número de individuos por los cuales un individuo es dominado.

Dependiendo del tipo de dominancia empleado se pueden clasificar en:

- Dominación de rango: El criterio se basa en por cuántos individuos es un individuo dominado. Ejemplos de algoritmos que entran en está clasificación son MOGA y NPGA.

- Dominación de conteo: El criterio se basa en cuántos individuos dominan un individuo. Ejemplos de algoritmos que entran en está clasificación son SPEA, SPEA2.
- Dominación de profundidad: El criterio se basa en que frente de dominancia está localizado el individuo (nichos). Ejemplos de algoritmos que entran en está clasificación son NSGA, NSGA-II.

5.5 Selección

La aproximación del conjunto Pareto óptimo está relacionada con dos posibles objetivos conflictivos: la distancia al frente óptimo debe ser minimizado y la diversidad de las soluciones generadas debe ser maximizado. En este contexto hay dos temas fundamentales cuando diseñamos un algoritmo evolutivo multiobjetivo: selección de apareamiento y la selección de entorno. El primero esta directamente relacionado con la cuestión de como guiarse a través del frente Pareto óptimo. Dado una piscina de individuos, los valores de *fitness* tiene que haber sido asignados con la bases de que individuos para la producción de descendencia son seleccionados. El procedimiento de llenar la piscina de apareamiento suele ser realizada aleatoriamente. El segundo aporta la pregunta de que individuos se mantienen durante el proceso de evolución. Debido al tiempo limitado y a los recursos de almacenamiento, solo una cierta fracción de individuos en una específica generación pueden ser copiados a la piscina de la siguiente generación. Es de práctica común usar una selección determinista para ello.
En la mayoría de EMO modernos estos dos conceptos son realizados del siguiente modo, aunque los detalles pueden ser diferentes. En principio, ambos métodos de selección son completamente independientes del otro:

- Selección de entorno: Además de la población, un archivo es mantenido que contiene una representación del frente no dominado entre todas las soluciones consideradas de lejos. Un miembro del archivo solo es eliminado si una solución ha sido encontrada que lo domina o el tamaño máximo del archivo es excedido y la porción del frente donde el miembro del archivo esta localizado es superpoblado. Normalmente, ser copiado al archivo es solo la única manera de que un individuo pueda sobrevivir a varias generaciones además de que la reproducción pura que puede ocurrir. Esta técnica es incorporada en orden de no perder ciertas porciones del actual frente no dominado debido a efectos aleatorios.
- Selección de apareamiento: La piscina de individuos en cada generación son evaluados en dos pasos del proceso. Primero todos los individuos son comparados en base de la relación de dominancia de Pareto, que define un orden parcial en este multiconjunto. Básicamente la información que con la cual individuos dominan a otros, es dominada por o es indiferente a ser usada para definir un *ranking* en la piscina de generación. Después este *ranking* es redefinido por la incorporación de información de densidad. Varias estimaciones de densidad son usadas para medir el tamaño del nicho donde el individuo específico esta localizado.

5.6 Aspectos de diseño avanzados

Además de estos fundamentos de diseño, junto con el elitismo que comentaremos luego, discutiremos dos aspectos más: manejo de constantes y articulación de preferencia.

En la optimización evolutiva monobjetivo han sido propuestas varias maneras de tratar con diferentes tipos de constantes (por ejemplo, la aproximación de la función de penalización). En principio, estas técnicas pueden ser utilizadas en presencia de múltiples criterios, pero la optimizacióm multiobjetivo suele ser más flexible al respecto. Una posibilidad es convertir cada constante en un objetivo separado, que tiene que ser optimizado además de los objetivos actuales. Alternativamente, las constantes pueden ser agregadas y solo un criterio de optimización -a minimizar la violación de constante general- es añadido. Ambos métodos tienen la ventaja de que no es necesaria ninguna modificación relacionada con el MOEA. Por otra parte, individuos no factibles que proveen buenos valores respecto a los actuales objetivos son tratados de igual manera que los individuos factibles con peores valores objetivo, que pueden volver lenta la velocidad de convergencia a través de la región factible. Métodos más sofisticados distinguen entre soluciones factibles y no factibles. Fonseca y Fleming sugirieron tratar cada constante como un objetivo distinto como anteriormente y extender la definición de Pareto dominancia en orden de favorecer la factibilidad en favor de

individuos no factibles. Si dos soluciones factibles son comprobadas para la dominancia, una es mejor que la otra si domina a su competidor con respecto a los objetivos actuales. Los mismos se mantienen si ambas soluciones son no factibles, sin embargo, la dominancia entonces es restrictiva solo con los objetivos constantes. Finalmente, soluciones factibles son definidas para dominar a las no factibles. Estos métodos aumenta la presión selectiva en el conjunto factible (asumiendo un asignamiento de *fitness* basado en Pareto). Una leve modificación de esta aproximación es considerar la violación de constante general en vez de tratar las constantes de forma separada, donde una solución no factible domina otra solución no factible si la violación de constante general es baja.

Las constantes representan un camino de incluir conocimiento existente sobre la aplicación en el proceso de optimización en orden de enfocarse en regiones prometedoras en la búsqueda de espacio. Además, otros tipos de información preferencial como los objetivos y sus *rankings* puede ser consultada y puede ayudar a guiar la búsqueda entre las regiones interesantes del conjunto Pareto óptimo. En instancia, Fonseca y Fleming extendieron la definición de la Pareto dominancia permitiendo añadir además de las constantes los objetivos y prioridades. Otra aproximación es el cambio de la definición de la Pareto dominancia para, aproximadamente hablando, considerar el punto general convexo de los conos en vez de los puntos que representan el área dominada de una solución dada.

5.7 1ª generación

Surgidos entre 1985-1995, la primera generación de algoritmos no emplean el conjunto de Pareto, por lo que no garantizan la obtención de la frontera de Pareto en su totalidad. Son eficientes y fáciles de implementar, aunque limitados para algunos pocos objetivos. La mayor dificultad de la optimización multiobjetivo es la aparición de objetivos conflictivos (Hans, 1988), es decir, ninguna de las soluciones factibles permite la solución óptima de todos los objetivos a la vez. Matemáticamente el óptimo de Pareto es la solución, ya que ofrece el menor conflicto entre objetivos. Excepto el VEGA que comentaremos a continuación, el resto se basan en el concepto de Pareto dominancia:

5.7.1 VEGA

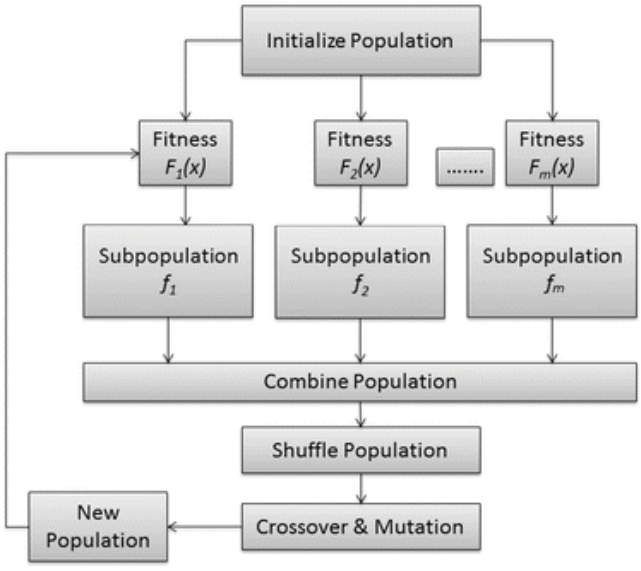


Figura 16: Diagrama de flujo del algoritmo VEGA

Conocido como *Vector Evaluated Genetic Algorithm* fue propuesto por Schaffer (1984) y se trata de una modificación del programa GENESIS de Grefenstette. Solo el mecanismo de selección del algoritmo genético es modificado de modo que en cada generación se genera una cantidad de subpoblaciones mediante una selección proporcional de acuerdo a cada función objetivo a la vez. Por lo tanto, para un problema de k objetivos y un tamaño de población M, se generarán k subpoblaciones de tamaño M/k cada una. Estas subpoblaciones se mezclarían para obtener una

nueva población de tamaño M, en la que el algoritmo genético aplicaría los operadores de cruce y mutación de forma habitual.
La ventaja de este método es que es eficiente y fácil de implementar, pero peca a veces de centrarse en algunos individuos o regiones. Si la selección proporcional es usada, entonces el mezclado y combinado de todas las subpoblaciones corresponden a la media del *fitness* de los componentes asociados a cada uno de los objetivos. En otras palabras, en esas condiciones, VEGA se comporta como una aproximación de agregación y por lo tanto está sujeto a los mismos problemas de esas técnicas.

Al ser un algoritmo que en ocasiones se centra en un grupo de individuos, Schaffer intentó minimizar esta especificación desarrollando dos heurísticas - la selección heurística de no dominancia (un esquema de buena redistribución) y la heurística de selección de pareja (esquema de cruzamiento). En la selección heurística de no dominancia, los individuos dominados son penalizados restando una penalización fija y pequeña de su esperado número de copias durante la selección. Entonces la penalización total para los individuos dominados seria dividida entre los individuos no dominados y seria añadida a su número de copias esperada durante la selección. Pero este algoritmo falla cuando la población tiene pocos individuos no dominados, dando un valor de *fitness* alto para esos pocos puntos no dominados, llevando a una alta presión selectiva. La heurística de selección de pareja fue implementada para promover el cruce de diferentes subgrupos. Esto fue implementado para seleccionar a un individuo, como pareja de una selección individual aleatoria, que tiene la distancia máxima euclídea en el espacio respecto a su pareja. Pero falla a prevenir la participación de los individuos más pobres en la selección de pareja, esto es debido a la selección aleatoria de la primera pareja y de la posibilidad de una distancia euclídea alta entre el ganador y el perdedor. Schaffer concluyo que la selección de pareja aleatoria es bastante superior que esta heurística.

5.7.2 MOGA

Conocido como *Multiobjective Optimization Genetic Algorithm* (Fonseca y Fleming, 1993), a cada

Data: $N', g, f_k(\mathbf{x})$
Result: N' miembros evolucionados g generaciones para solucionar $f_k(\mathbf{x})$
begin
  Inicializa la población P';
  Evalúa los valores objetivo;
  Asignación de ranqueo basado en la dominancia Pareto;
  Computar el conteo de nichos;
  Asignar el fitness escalado lineal;
  Compartir el fitness;
  for *i=1* to g **do**
    Selección vía torneo estocástico universal;
    Cruce de un punto;
    Mutación;
    Evaluar los valores objetivo;
    Asignar ranqueo basado en la Pareto dominancia;
    Calcular el conteo de nichos;
    Asignamos fitness linealmente escalado;
    Asignamos fitness sharing;
  end
end

Algoritmo 4: MOGA

individuo de la población se le asigna un rango de acuerdo al cual será ordenado para la selección (Goldberg, 1989). La complejidad computacional $O(kM^2)$ dada por la necesidad de revisar la no dominancia, donde k es la cantidad de objetivos y M es el tamaño de la población. Nuevas versiones emplean *fitness sharing* y restricciones para los cruces, formas y continuidad del frente de Pareto, además de que utilizan técnicas de proporción de nichos sobre los objetivos. Fue el primer MOGA verdadero, superando a sus competidores contemporáneos, además de que ha sido combinado con redes neuronales para mejorar su rendimiento. El desempeño depende de una buena selección y el factor de compartición.

Los pasos del algoritmos son los siguientes: primero se ordena de menor (mejor) a mayor la población acorde al rango que se le ha asignado a cada individuo mediante un criterio de no dominancia. Vale 1 si ningún individuo lo domina, en otro caso hay que sumarle a 1 el número de individuos que lo dominan; en el segundo paso se asigna el valor de adaptación para cada individuo mediante interpolación lineal de alguna función (Goldberg) desde el mejor hasta el peor; en el último paso se promedia la adaptación de los individuos del mismo rango para que todos tengan el mismo valor de adaptación. Este procedimiento mantiene el valor de aptitud de la población constante mientras mantiene la presión selectiva apropiada, debido a que Goldberg y Deb indicaron que este tipo de asignamiento de *fitness* puede producir convergencia prematura. Para evitarlo, se ha usado un método de formación de nichos para distribuir a los individuos en la región óptima de Pareto, pero en vez de realizar el *fitness sharing* en los valores de parámetros, lo han hecho con los valores de la función objetivo.

5.7.3 NSGA

Propuesto por Srinivas y Deb (1994) y posteriormente revisado en Deb (1999), conocido como *Nondominated Sorting Genetic Algorithm*, la idea detrás de este método es un método de selección de ranqueo usado para destacar buenos puntos y un método de nichos (dominación en profundidad) es usado para mantener subpoblaciones estables de buenos puntos. NSGA varia del algoritmos genético simple solo en como funciona el operador de selección. Los operadores de cruce y mutación se mantienen iguales. El operador de selección se realiza mediante sobrante estocástico. Las

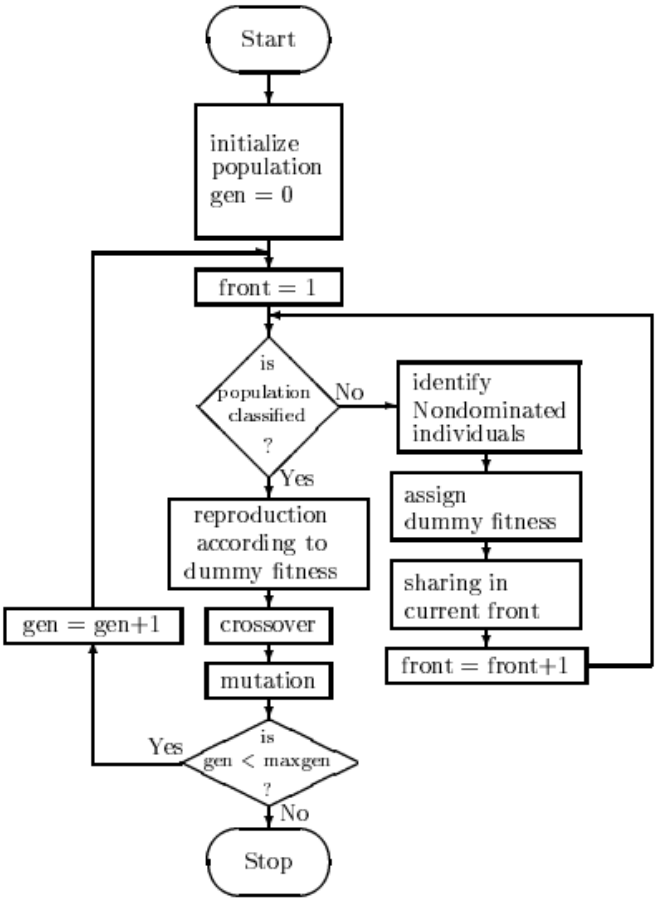


Figura 17: Diagrama de flujo del algoritmo NSGA

características más significativas del algoritmo son:

- Convergencia rápida de la población hacia las regiones no dominadas.
- El procedimiento para compartir *fitness* (*fitness sharing*) (método de proporción continua en el fenotipo) ayuda a distribuir las soluciones encontradas a lo largo de todo el ámbito de

```
Data: N', g, f_k(x)
Result: N' miembros evolucionados g generaciones para solucionar f_k(x)
begin
    Inicializamos la población P';
    Evaluamos los valores objetivo de cada individuo;
    Asignamos ranqueo en nichos basados en la Pareto dominancia hasta que no quede
     ninguno por ranquear;
    Computamos el conteo de nichos;
    Asignamos fitness compartido;
    for i=1 to g do
        Selección via sampleo universal estocástico;
        Cruzamiento de 1 punto;
        Mutación;
        Evaluamos los valores objetivo de cada individuo;
        Asignamos ranqueo en nichos basados en la Pareto dominancia hasta que no quede
         ninguno por ranquear;
        Computamos el conteo de nichos;
        Asignamos fitness compartido;
    end
end
```

Algoritmo 5: NSGA

búsqueda.

- No tiene límites en cuanto a la cantidad de funciones objetivo a optimizar.
- Puede tratar tanto con criterios de maximización/minimización.

Este algoritmo no es elitista, aunque se puede añadir. Antes de que la selección sea realizada, la población es ranqueada mediante nichos con conjuntos de poblaciones Pareto. Las soluciones no dominados son identificados como la actual población. Entonces, todos ellos constituyen en primer frente de población y se les asigna un valor muy grande ficticio. El mismo valor de *fitness* es asignado para dar una reproducción potencial equitativa a todos esos individuos no dominados. Dependiendo del problema, el contador de nicho puede fijarse en la similitud entre los genes o los cromosomas, teniendo:

- En el primer caso se está distribuyendo la población de modo que los individuos resultantes tengan codificaciones bien diferenciadas unos de otros.
- En el segundo caso, la distribución permite que los individuos resultantes sean bien diferentes, sin importar la codificación de los mismos.

La técnica más conocida para mantener soluciones repartidas de forma uniforme en el frente de Pareto en los algoritmos genéticos multiobjetivo es la distancia entre dos soluciones y está distancia se puede representar como la distancia euclídea de estas soluciones en el espacio objetivo.

Data: P
begin
  $rango_{maximo} = 0$;
  for $i = 1$ to $i = N$ **do**
    Calcular rango r_i del individuo x_i;
    if $rango_{maximo} < r_i$ **then**
      $rango_{maximo} = r_i$;
    end
  end
  $FitVir = N$;
  $RanAc = 1$;
  for $i = 1$ to $RanMax$ **do**
    Asignar $FitVir$ como valor de fitness virtual a los miembros de P que tengan rango i;
    Compartir fitness virtual entre los individuos de rango i para obtener los valores de fitness reales;
    $FitVir$ = Mínimo entre todos los fitness reales asignados - ε_{niv};
  end
end

Algoritmo 6: Asignamiento de fitness NSGA

Para intentar mantener la diversidad en la población, estos individuos clasificados son compartidos con su valor de *fitness* ficticio (*fitness sharing*). La compartición se realiza en la selección usando valores de aptitud degradados que son obtenidos al dividir el *fitness* original entre la cantidad proporcional de número de individuos alrededor de él (dando como parámetro α para el *fitness sharing* el valor 2). Según Deb, el NSGA no es muy sensible a este parámetro. Esto causa que coexistan múltiples puntos óptimos en la población. Después de la compartición, esos individuos no dominados son ignorados temporalmente para procesar el resto de la población con el mismo método para identificar a los individuos en el segundo frente no dominado. Este nuevo conjunto de puntos son asignados con un nuevo valor de aptitud ficticio que es menor que el mínimo *fitness sharing* ficticio del frente previo. El proceso continua hasta que la población es ordenada en varios frentes. La población entonces se reproduce de acuerdo al valor de aptitud ficticio.

Posteriormente se realiza la selección mediante un sistema de selección proporcional al valor de aptitud o sistema de ruleta para orientar la búsqueda hacia regiones no dominadas. Por lo tanto, los individuos del primer frente tienen más probabilidades de ser escogidos que el resto de los frentes. El operador de cruce se realiza con una probabilidad $p_c = 1$. El operador de mutación solo se realiza a un porcentaje de hijos y a cada gen de cada hijo elegido se le aplica una probabilidad de mutación de r_m.

Las ventajas de este algoritmo es que la diversidad entre las soluciones no dominadas es mantenida mediante el procedimiento *crowding* sin necesitar ningún otro control de diversidad. Como desventajas de este algoritmo se encuentran que no implementa el concepto de elitismo, ya que las soluciones finales surgen de los individuos no dominados de la última generación, además que al pasar las generaciones se suelen perder soluciones muy buenas y no tiene porque reemplazarse por otras mejoras. Además, el resultado del algoritmo no suele dar muchos individuos que formen parte del conjunto Pareto real y tiende a finalizar con una población muy homogenia. A pesar de que se trata de un algoritmo muy efectivo, también es criticado por su complejidad computacional y por tener que escoger *a priori* la elección del parámetro σ_{share}.

5.7.4 NPGA/NPGA 2

Conocido como *Niched Pareto Genetic Algorithm* (Horn y Nafpliotis, 1993), este algoritmo se basa en dos conceptos:

- Concepto de Pareto dominancia.
- Combinación de soluciones de la selección de torneo.

Dados dos individuos que compiten entre sí, se selecciona un subconjunto aleatorio de la población de tamaño t_{dom}, que suele ser del 10 %. Si un único individuo de los dos no es dominado por ningún miembro del subconjunto, entonces se convierte directamente en el ganador del torneo. Si los dos elementos son dominados o no dominados (empate), el torneo se decide por *fitness* compartido en el dominio del objetivo, utilizando una técnica llamada clase equivalente compartida (método de proporción). El individuo con menos individuos en su nicho es escogido. Entre sus ventajas e inconvenientes están que no aplica el *ranking* Pareto a la población externa (eficiencia) además de que requiere de un parámetro adicional (tamaño del torneo) dado el factor de compartición.

Existe una segunda versión denominada NPGA 2 (M. Erickson), algoritmo que hace uso de *ranking* Pareto manteniendo la selección por torneo (similar al NPGA). En este algoritmo se utilizan individuos de la próxima generación (población parcialmente completada). Esta técnica se conoce como actualización de *fitness sharing* continua, propuesta por Oei et al.

```
Data: N', g, f_k(x)
Result: N' miembros evolucionados g generaciones para solucionar f_k(x)
begin
    Inicializar población P;
    Evaluar valor objetivo;
    for i=1 to g do
        Selección mediante torneo binario usando ranqueo como grado de dominación;
        if candidato 1 dominado then
            Seleccionar candidato 2;
        else if candidato 2 dominado then
            Seleccionar candidato 1;
        else
            Realizar fitness sharing;
            Devolver candidato con menor conteo de nicho;
        Cruce de un punto;
        Mutación;
        Evaluar valores objetivo;
    end
end
```

Algoritmo 7: NPGA 2

5.7.5 Otros

Algunos algoritmos conocidos que no utilizan el concepto del óptimo de Pareto y están basados en los métodos clásicos de los problemas de optimización multiobjetivo comentados anteriormente son los siguientes:

- *Vector optimized ES*
- *Weight based Genetic Algorithm*
- VOW-GA: *Variable Objectie Weighting Genetic Algorithm* (Hajela y Lin, 1992).
- RW-GA: *Random Weights Genetic Algorithm* (Ishibuchi y Murata, 1998).
- *Adaptive Weight Genetic Algorithm* (Gen y Cheng, 2000).
- *Interactive Adaptive Weight Genetic Algorithm* (Lin y Gen, 2008).

Y otros basados en Pareto:

- Ranking Pareto puro (Goldberg, 1989)
- PDE: *Pareto-Frontier Differential Evolution Algorithm* (2001)

5.8 Elitismo

El elitismo tiene como finalidad no perder las mejores soluciones encontradas durante la ejecución del algoritmo genético. En algunos algoritmos estos individuos pueden perderse por motivos aleatorios, por lo que este concepto permite tratar de un modo más efectivo el frente de Pareto.

Para no perder buenos individuos, uno de los mecanismos a utilizar consistiría en generar la población de la siguiente generación a partir de la antigua población y sus descendientes, en lo que se conoce como piscina de cruce. Un ejemplo de algoritmo que utiliza este mecanismo es el NSGA-II.

Por otro lado se puede añadir una población auxiliar conocida como *archive* (archivo). La finalidad del archivo es mantener un conjunto de soluciones extras separado del motor de optimización, que pueden ser incorporadas a los procesos de selección. Estas dos variantes son ilustradas en la siguiente figura.

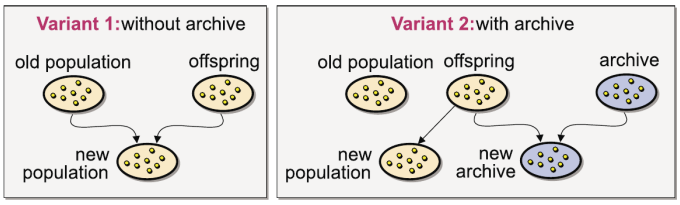


Figura 18: Dos métodos de implementación de elitismo

Normalmente el tamaño del archivo es restrictivo debido a la memoria, al igual que a las limitaciones de ejecución. Por lo tanto, hay que definir un criterio básico con el cual seleccionar las soluciones que van a estar dentro del archivo. El criterio de dominancia es comúnmente usado. Si un archivo es utilizado, el archivo comprende solo la actual aproximación del conjunto de Pareto. Sin embargo, el criterio de dominancia por norma general no es suficiente (por ejemplo, para problemas continuos, el conjunto de Pareto puede contener un número infinito de soluciones), por lo tanto, se tiene que tener en cuenta información adicional para reducir el número de soluciones guardadas, como por ejemplo, la densidad y el tiempo que ha pasado desde que el individuo ha entrado en el archivo.

Muchos MOEA hacen uso de tanto de la dominancia como de la densidad para escoger a los individuos que tienen que estar en el archivo en la siguiente generación. El espacio es dividido a través de zonas cuadradas y se cuentan cuantos individuos hay dentro de cada uno. Esto permite calcular la densidad de individuos en un cuadrado, sin embargo estas aproximaciones pueden sufrir de problemas de deterioración, por ejemplo, soluciones contenidas en el archivo en la generación actual pueden ser dominadas por soluciones que eran miembros del archivo en cualquier generación anterior y fueron descartadas anteriormente. Laumanns et al. presentó una estrategia de archivado que evita este problema y garantiza el mantener un conjunto diverso de soluciones Pareto óptimas.. Los diferentes MOEA elitistas difieren, entre otras cosas, en la aplicación de estas tres estrategias, que son:

- Estrategia de elitismo: Modo en el que la población elitista se actualiza.
- Estrategia de evaluación: Modo en el que los individuos del conjunto élite afectan a la asignación de *fitness* en la población y viceversa.
- Estrategia de reinserción: Modo en el que los individuos toman parte en el proceso de reproducción de descendientes.

5.9 Hipervolumen

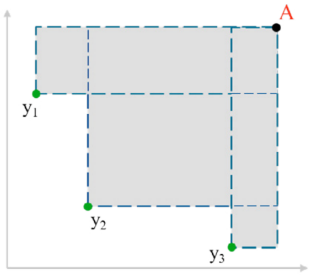
Esta métrica representa el volumen del espacio objetivo que es cubierto por los individuos de un conjunto de soluciones no dominadas (pertenecen al frente de Pareto). Sirve como métrica para determinar la dominancia de un algoritmo frente a otros de acuerdo a los resultados obtenidos. El volumen es delimitado por dos puntos: un punto que es conocido como el punto antióptimo (A) que es definido como la peor solución dentro del espacio objetivo, y un segundo punto óptimo (pseudoóptimo) que es calculado por el propuesto método de solución. Una normalización de los valores objetivo es requerida antes de calcular el hipervolumen. El hipervolumen es la superficie

(volumen), dependiendo del número de funciones objetivo, que está dentro de los individuos $D_l(l = 1, ..., \Upsilon)$ del conjunto no dominado de soluciones (ND) y el punto A, como se ve en la siguiente ecuación:

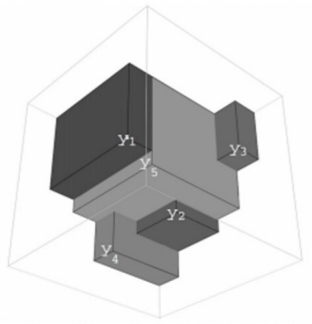
$$\boxed{HV(ND, A) = \wedge(\{\bigcup h(D_l)|D_l \in ND, l = 1, ..., \Upsilon\})} \quad (19)$$

Donde $h(D_l) = |D_{l1} - A_1| \times |D_{l2} - A_2| \times ... \times |D_{lC} - A_C|$, Υ es el número de soluciones que están en el frente de Pareto y C es el número de funciones objetivo.

En la siguiente figura, la superficie es obtenida por la unión de diferentes rectángulos creados por cada solución del frente (y_1, y_2 y y_3) y el punto de referencia (A). Entonces, la siguiente figura muestra el cálculo del hipervolumen con tres funciones objetivo donde hay cinco soluciones en el frente (y_1, y_2, y_3, y_4 y y_5) y un punto de referencia.



(a) Problema dos objetivos



(b) Problema tres objetivos

Figura 19: Ejemplos hipervolumen

Aunque en un principio se estudió este concepto para usarlo como métrica de comparación de algoritmos, empezó a ser utilizado para el paso de selección de los algoritmos genéticos multiobjetivo como el PESA II o el NSGA-III y los algoritmos genéticos multiobjetivo más recientes los utilizan por lo que podría tratarse del próximo paradigma de los algoritmos genéticos multiobjetivo para dar paso a la 3ª generación.

5.10 2ª generación

Luego nos encontramos con la segunda generación, entre cuyas características enfatizan en la eficiencia y en que se usan poblaciones secundarias (o externas) para generar soluciones que sean no dominadas y uniformemente distribuidas (elitismo). Los problemas que se tratan en esta generación suelen utilizar *fitness sharing* con complejidad $O(M^2)$ y la verificación de la dominancia de Pareto con complejidad $O(kM^2)$. Varios ejemplos de esta generación son los siguientes:

5.10.1 NSGA-II

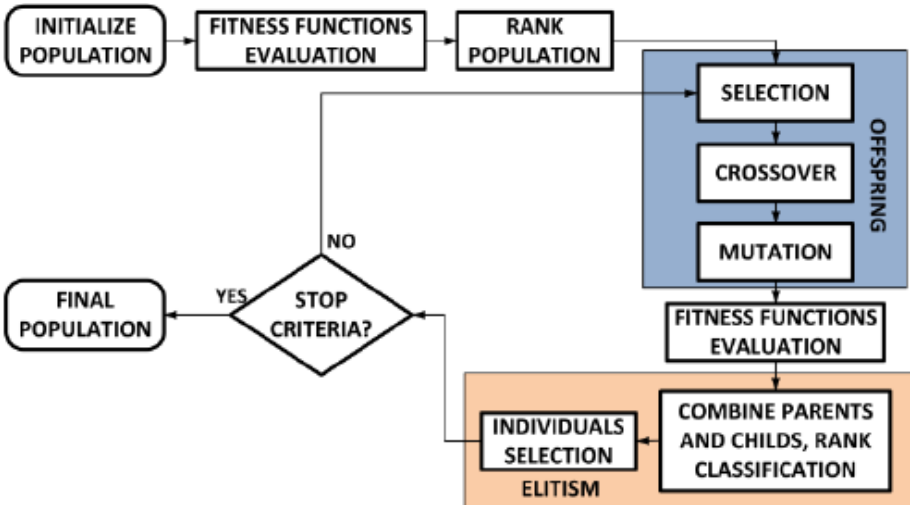


Figura 20: Diagrama de flujo del algoritmo NSGA-II

Considerado el mejor modelo paralelo, (aunque en ejecución secuencial el SPEA2 suele dar mejores resultados a costa de un coste computacional mayor) *Elitist Non-Dominated Sorting Genetic Algorithm* fue propuesto por Deb y sus estudiantes, 2000.

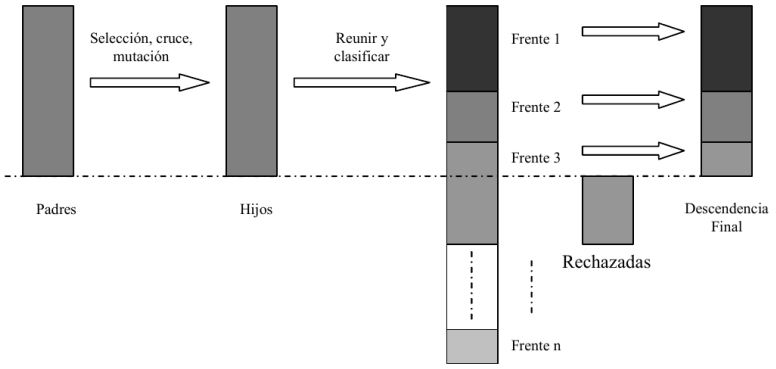


Figura 21: Procedimiento NSGA-II

La estrategia para garantizar la diversidad es poco relevante en las primeras generaciones, ya que en estas etapas existen muchos frentes que sobreviven al paso de las generaciones, pero a medida que el algoritmo progresa muchos individuos van a formar parte del mejor frente, incluso haciendo que dicho frente se tenga que cribar, por lo que se hace importante que las soluciones no rechazadas sean buenas y escogidas mediante un método que garantice la diversidad dentro de un frente de dominancia.

El algoritmo utilizado para asignar los frentes de dominancia de la población es el siguiente, donde 1 es el mejor nivel:

begin
  for p en P **do**
    Inicializar $S_p = \emptyset$. Esto contiene todos los individuos que son dominados por p;
    Inicializar $n_p = 0$. Esto debería ser el número de individuos que domina p;
    for q en P **do**
      if p *domina* q **then**
        Añadir q al conjunto S_p;
      else
        if q *domina* p **then**
          Incrementar el conteo de dominación para p ($n_p = n_p + 1$);
        end
      end
    end
    Si $n_p = 0$, no hay individuos que dominen p, entonces p pertenece al primer frente ($p_{rank} = 1$). Actualizar el primer conjunto de frente añadiendo p al frente 1 ($F_1 = F_1 \cup \{p\}$);
  end
  Inicializar el conteo del frente a 1 ($i = 1$);
  while $F_i \neq \emptyset$ **do**
    $Q = \emptyset$. El conjunto para guardar individuos para el frente $(i + 1)$;
    for $p \in F_i$ **do**
      for $q \in S_p$ *(S_p es el conjunto de individuos dominado por p)* **do**
        $n_q = n_q - 1$;
        if $n_q == 0$ *(q pertenece al siguiente frente)* **then**
          $q_{rank} = i + 1$;
          $Q = Q \cup \{q\}$
        end
      end
    end
    $i = i + 1$;
    $F_i = Q$;
  end
end

Algoritmo 8: Ordenamiento de no dominancia NSGA-II

Los principales pasos del algoritmo son los siguientes: el primer paso genera una población P de N tamaño; el segundo paso consiste en ordenar la población mediante Pareto dominancia y se evalúan las distancias de apilamiento para todos los individuos de la población; en el tercer paso se generan N descendientes mediante selección, cruzamiento y mutación; el cuarto paso consiste en reunir padres y descendientes (con un tamaño de $2N$) y clasificarlos mediante frentes de Pareto dominancia; el quinto paso consiste en determinar los individuos que pasarán a la siguiente generación seleccionando los mejores frentes. En caso de que se superase el límite de tamaño N, se deberá proceder a seleccionar las soluciones de mayor distancia de apilamiento del último frente que ha sido tratado. La condición de parada consiste en que si no se cumple, hay que volver al tercer paso, en caso contrario se procede a terminar la ejecución del algoritmo. La selección por torneo para el apilamiento compara dos soluciones y elige un individuo como ganador. Cada solución tiene un rango de no dominancia asociado (r_i) y también tiene que tener una distancia de apilamiento (d_i) asignada. La distancia de apilamiento d_i de una solución i es la distancia en el espacio objetivo alrededor de i que no esté ocupado por otro individuo de la población. A través de estos dos atributos se puede encontrar los mejores individuos primero a través de aquel individuo que tenga mejor rango. Como hemos comentado anteriormente, en el caso de estar situadas en el mismo frente de dominancia, aquella solución a que tenga mejor (más alta) distancia de apilamiento es la ganadora del torneo. La distancia de apilamiento $d_{I_j^m}$ para cada solución j, según un índice I, es determinada mediante la siguiente ecuación

$$\boxed{d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{max} - f_m^{min}}} \tag{20}$$

donde f_m^{max}, f_m^{min} son el máximo y mínimo valor de la función objetivo m respectivamente, $f_m^{(I_{j+1}^m)}$, $f_m^{(I_{j-1}^m)}$ son los individuos vecinos al individuo j para cada una de las funciones objetivo m. La distancia considera todos los objetivos y para garantizar la máxima diversidad los individuos que conforman el límite del frente de dominancia siempre son escogidos asignándoles un valor infinito. La distancia resultante es la suma de las distancias en cada una de las direcciones de las funciones objetivo del problema. La idea básica es encontrar la distancia euclídea entre cada individuo en el frente de dominancia correspondiente.

De este modo, el algoritmo promueve hacia el siguiente ciclo generacional los individuos que ocupen los mejores frentes, y dentro de estos escoge los individuos más diversos a través de las distancias de apilamiento en caso de que se tenga que cribar el frente. Para el operador de cruce, Deb recomienda utilizar el *Simulated Binary Crossover* (SBX). Este simula el cruce binario observado en la naturaleza, expresado matemáticamente:

$$\boxed{\begin{aligned} c_{1,k} &= \frac{1}{2}\left[(1-\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}\right] \\ c_{2,k} &= \frac{1}{2}\left[(1-\beta_k)p_{1,k} + (1-\beta_k)p_{2,k}\right] \end{aligned}} \tag{21}$$

donde $c_{i,k}$ es el i-ésimo hijo con k-ésimo componente, $p_{i,k}$ es el padre seleccionado y $\beta_k(\geq 0)$ es una muestra de un número generado aleatoriamente teniendo la densidad

$$\boxed{\begin{aligned} p(\beta) &= \frac{1}{2}(\eta_c+1)\beta^{\eta_c}, && \text{si } 0 \leq \beta \leq 1 \\ p(\beta) &= \frac{1}{2}(\eta_c+1)\frac{1}{\beta^{\eta_c+2}}, && \text{si } \beta > 1 \end{aligned}} \tag{22}$$

Esta distribución puede ser obtenida de la generación de un número aleatorio uniformemente u entre (0, 1). η_c es el índice de distribución del cruce (esto determina cuan bien está distribuido el hijo respecto a sus padres), es decir

$$\boxed{\begin{aligned} \beta(u) &= (2u)^{\frac{1}{(\eta+1)}} \\ \beta(u) &= \frac{1}{[2(1-u)]^{\frac{1}{(\eta+1)}}} \end{aligned}} \tag{23}$$

Para el operador de mutación se utiliza la mutación polinomial

$$\boxed{c_k = p_k + (p_k^u - p_k^l)\delta_k} \tag{24}$$

donde c_k es el hijo y p_k es el padre con p_k^u empezando por el límite superior en el componente del padre, p_k^l es el límite inferior y δ_k es una pequeña variación que es calculada con una distribución polinomial usando

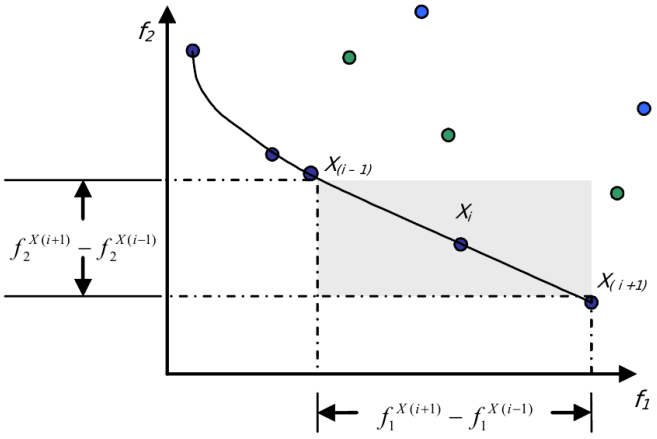
$$\boxed{\begin{aligned} \delta_k &= (2r_k)^{\frac{1}{\eta_m+1}} - 1, && \text{si } r_k < 0{,}5 \\ \delta_k &= 1 - [2(1-r_k)]^{\frac{1}{\eta_m+1}}, && \text{si } r_k \geq 0{,}5 \end{aligned}} \tag{25}$$

donde r_k es un número generado aleatoriamente uniformemente entre (0, 1) y η_m es el índice de distribución para la mutación.

Han habido varias propuestas alternativas para modificarlo según que necesidad (reducir el coste computacional, ...). Una posible modificación es la de importar la lógica del algoritmo genético de Chu-Beasley (CBGA) para evitar las N evaluaciones de la función objetivo en la población en cada evaluación. Las ventajas de esta versión respecto a su antecesor es que tiene un algoritmo de ordenado más eficiente computacionalmente, incorpora elitismo y no utiliza el *fitness sharing* que debe ser escogido *a priori*. Además es altamente competitivo en convergencia de Pareto. Inconvenientes que tiene son que parece no rendir bien cuando la población está representada mediante una representación binaria y tiende a tener problemas exploratorios a medida que se incrementa el número de objetivos (la mayoría de algoritmos actuales acusan este último problema).

```
crowding-distance-assignment(I)
l = |I|                                          number of solutions in I
for each i, set I[i]_distance = 0                initialize distance
for each objective m
  I = sort(I, m)                                 sort using each objective value
  I[1]_distance = I[l]_distance = ∞              so that boundary points are always selected
  for i = 2 to (l − 1)                           for all other points
    I[i]_distance = I[i]_distance + (I[i+1].m − I[i−1].m)/(f_m^max − f_m^min)
```

(a) Algoritmo



(b) Representación gráfica

Figura 22: Distancia de apilamiento

5.10.2 PESA/PESA II

Conocido como *Pareto Envelope-based Selection Algorithm*, usa una población interna pequeña con individuos candidatos a entrar en el archivo, normalmente igual de grande como en el SPEA. Como en el PAES, emplea una división del *hypergrid* del espacio del fenotipo (es decir, la función objetivo) para mantener la diversidad y permite realizar un seguimiento del grado de cruce en diferentes regiones del archivo. La selección se basa en una medida de agrupamiento de los *grids*, es decir, en una medida de cruce. Con esta medida se decide cuál solución se colocará en la población externa que almacenará las soluciones no dominadas encontradas durante el proceso. Como en PAES y SPEA, el reemplazamiento (decidir que tiene que dejar el archivo si se llena) también esta basada en una medida de cruce.

Aparte de los parámetros estándar como el cruce y la mutación, PESA tiene dos parámetros relacionados con el tamaño de la población, y un parámetro relacionado con la estrategia de cruce *hypergrid*. Una descripción detallada del algoritmo es la siguiente, con dos parámetros de población P_I (el tamaño de la población interna) y P_E (el tamaño máximo del archivo o 'población externa').

1. Genera y evalúa cada de la población inicial 'interna' (IP) de P_I cromosomas, y inicializa la población 'externa' (EP) con el conjunto vacío.

2. Incorpora los miembros no dominados de IP a EP.

3. Si un criterio de parada ha sido alcanzado, entonces para, devolviendo el conjunto de cromosomas en EP como resultado. Si no, borra el actual contenido de IP, y repite lo siguiente hasta que las nuevas soluciones candidatas de P_I han sido generadas:

   - Con probabilidad p_c, seleccionamos dos padres de EP, producimos un único hijo vía cruce, y mutamos al hijo. Con probabilidad $(1 - p_c)$, seleccionamos un padre y lo mutamos produciendo un hijo

4. Volvemos al paso 2.

En el paso de 'incorporación en el archivo' (paso 2), el actual conjunto de nuevos candidatos a ser soluciones (IP) son incorporados al archivo de uno en uno. Un candidato puede entrar al archivo si no es dominado dentro del IP, y si no es dominado por un miembro actual del archivo. Una vez un candidato ha entrado en el archivo, cualquier miembro del archivo que sea dominado será

sacado del archivo. Si el añadir un candidato muestra el archivo lleno (su tamaño temporalmente se convierte en $P_E + 1$), entonces un miembro actual de EP es eliminado. La elección de ese miembro eliminado se detallará luego.

La selección de un padre en PESA, dentro del paso 3, esta basado en el grado de cruce en diferentes regiones del archivo. Esto es ilustrado en la siguiente figura. En la figura, un número de puntos son vistos en el espacio fenotipo de un problema de minimización biobjetivo. Los círculos son puntos no dominados, y por lo tanto están en el archivo. Los cuadrados son dominados por los miembros del archivo, pero pueden ser puntos en la actual población interna. La estrategia de

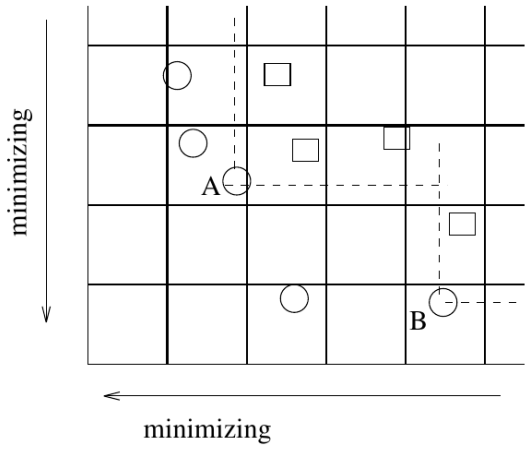


Figura 23: PESA crowding

cruce en PESA funciona con un *hypergrid* implícito que divide (normaliza) el espacio fenotipo en hipercajas. En la figura anterior esto es ilustrado en las gruesas líneas horizontales y líneas verticales; el problema es bidimensional y por lo tanto estas hipercajas son simples cuadrados. Cada cromosoma en el archivo es asociado con una hipercaja particular en el espacio fenotipo, y tiene un atributo llamada 'factor de exprimir', que es simplemente el número total de otros cromosomas en el archivo que habitan la misma caja. Por ejemplo, el factor de exprimir del cromosoma A en la figura anterior es 2, y el del cromosoma B es 1. El factor de exprimir es usado para el *fitness* selectivo. Por ejemplo, cuando PESA usa un torneo de selección binaria, dos cromosomas son elegidos de forma aleatoria del archivo, y el que tenga menor factor de exprimir es escogido (rompiendo los lazos de forma aleatoria), por lo tanto los esfuerzos para orientar la búsqueda hacia áreas el frente de Pareto emergente que actualmente tiene una pequeña representación en la población.

El factor de exprimir es también usado para la actualización del archivo. Como es indicado abajo, la incorporación de un miembro de la población interna en el archivo puede llevar a que el tamaño del archivo exceda el máximo tamaño de E_p temporalmente. Una solución debe ser ser eliminada del archivo. La elección es hecha encontrando el máximo factor de exprimir en la población, y quitando un cromosoma arbitrario que tenga ese factor de exprimir.

Como hemos visto, PESA usa el factor de exprimir en las dos de estas estrategias. En el paso de selección del PAES, ya que PESA usa actualmente un algoritmo de escalada, la selección solo es realizada entre dos candidatos: la solución actual, y una mutante. Si el archivo esta lleno, el factor de exprimir está en realidad usado en PAES, y si el mutante tiene un factor de exprimir más bajo que el actual, entonces pasa a ser la nueva solución. Hasta cierto punto PAES es bastante similar a PESA en emplear el factor de exprimir tanto en la selección como en la actualización del archivo, sin embargo el hecho que PAES es un método de búsqueda local y PESA es una población basada en una técnica de renderizado diferente al resto de algoritmos.

Hay una revisión llamada PESA II que en vez de una asignación selectiva de *fitness* (*individual-based selection*) como el NSGA-II o el SPEA, la selección de *fitness* es asignada a hipercajas en el espacio objetivo (*region-based selection*) que actualmente está ocupada por mínimo un individuo en la aproximación actual del frente de Pareto. Una hipercaja es de este modo seleccionada, y el resultado de la selección individual es elegida de forma aleatoria en esa hipercaja. Este método de

selección es visto como más sensible a asegurar una buena dispersión de desarrollo sobre el frente de Pareto que la selección basada en el individualismo. El método es implementado en un algoritmo evolutivo multiobjetivo moderno. La nueva técnica de selección es encontrada significativamente superior a otros métodos como PAES, PESA y SPEA.

La selección basada en la región provee una alternativa en la cual el objetivo es alcanzado más directamente. Utilizando un método de selección estándar luego de seleccionar una hipercaja, un individuo es escogido aleatoriamente para realizar las operaciones genéticas. El siguiente análisis aporta porque la selección basada en la región puede ser mejor que un esquema basado en el individuo. Asumiremos que usamos una selección de torneo binaria. Supongamos que tenemos dos hipercajas donde uno está ocupado por 9 individuos y el otro por un único individuo que conforman la aproximación actual del frente de Pareto. Con la selección de torneo binaria, la posibilidad de seleccionar el mejor individuo en una selección basada en el individuo sería $1 - (9/10)^2 = 0{,}19$ mientras que la posibilidad de escoger los 9 restantes es de 0.81. Esto no parece proveer una alta parcialidad en las regiones menos llenas. Con la selección basada en regiones, sin embargo, las unidades de selección son las dos hipercajas ocupadas. La probabilidad de elegir el individuo más aislado (y por lo tanto el mejor individuo) es $1 - (1/2)^2 = 0{,}75$. Por lo tanto, la selección basada en individuos suele eligir a individuos menos aislados, mientras que la selección basada en regiones es justo lo contrario.

Consideremos una aproximación del frente de Pareto que tiene b hipercajas ocupadas, con n_i individuos en la caja i y P individuos que son ocupados por hipercajas, por lo tanto $\sum_{i=1}^{b} n_i = P$. Asumamos con una pequeña perdida de generalidad que una hipercaja j tiene el b_i más grande y otra hipercaja tiene el menor n_i. El número de individuos en estos menor y mayor hipercajas son l y m respectivamente.

Cuando usamos la selección basada en individuos, la oportunidad de elegir un individuo en la menor hipercaja será de $1 - ((P - l)/P)^2$. Para la hipercaja mayor es $(m/P)^2$; el ratio de esas probabilidades es $(2Pl - l^2)/m^2$. Cuando m es alto respecto a l, la posibilidad relativa de escoger un individuo aislado respecto a un individuo rodeado se reduce drásticamente, por lo que parecería poco razonable centrarse en las regiones pobladas. En contraste, el ratio correspondiente para la selección basada en la región se vuelve se convierte en $2b - 1$. Esto último no se ve afectado por el número relativo de individuos en las diferentes cajas, y nunca es menor que 1 (de hecho, siempre es menor que 3 cuando hay más de una hipercaja ocupada).

Esto daría a pensar que el efecto de centrarse en regiones aisladas que en la pobladas con una selección basada en los individuos con un torneo de tamaño elevado. Sin embargo, hay que darse cuenta que la posibilidad de escoger un individuo de la caja más poblada en este caso será de $(m/P)^k$, donde k es el tamaño del torneo. Cuando el tamaño del torneo es alto, esto suelta una población muy numerosa, y en esas condiciones, la oportunidad de elegir un individuo de la caja menos poblada se vuelve inaceptablemente baja, afectando a la capacidad de exploración del algoritmo.

Respecto a la complejidad computacional de las dos aproximaciones de selección, suponiendo que por cada generación se genera una nueva población de tamaño n, la selección basada en el individuo tiene que estimar el grado de 'aislamiento' de cada individuo. En una población de tamaño n debería requerir n^2 operaciones de comparación, donde las distancias entre todas las distintas parejas es calculado. Sin embargo, en NSGA-II y PAES, por ejemplo, esto se puede realizar más rápido. El NSGA-II requiere $\Theta(k * n\log\, n)$ de tiempo computacional, donde k es el número de objetivos. En la selección basada en la región, la clave computaciones es calcular el ID de la hipercaja para cada individuo. Como es indicado en Knowles y Corne (2000), en un problema de k objetivos usando un *grid* de g^k hipercajas, solo $\Theta(k * n)$ comparaciones son hechas por generación. La eficiencia es mejorada si g es una potencia de 2, pero la complejidad espacial es lineal en n.

5.10.3 SPEA/SPEA2

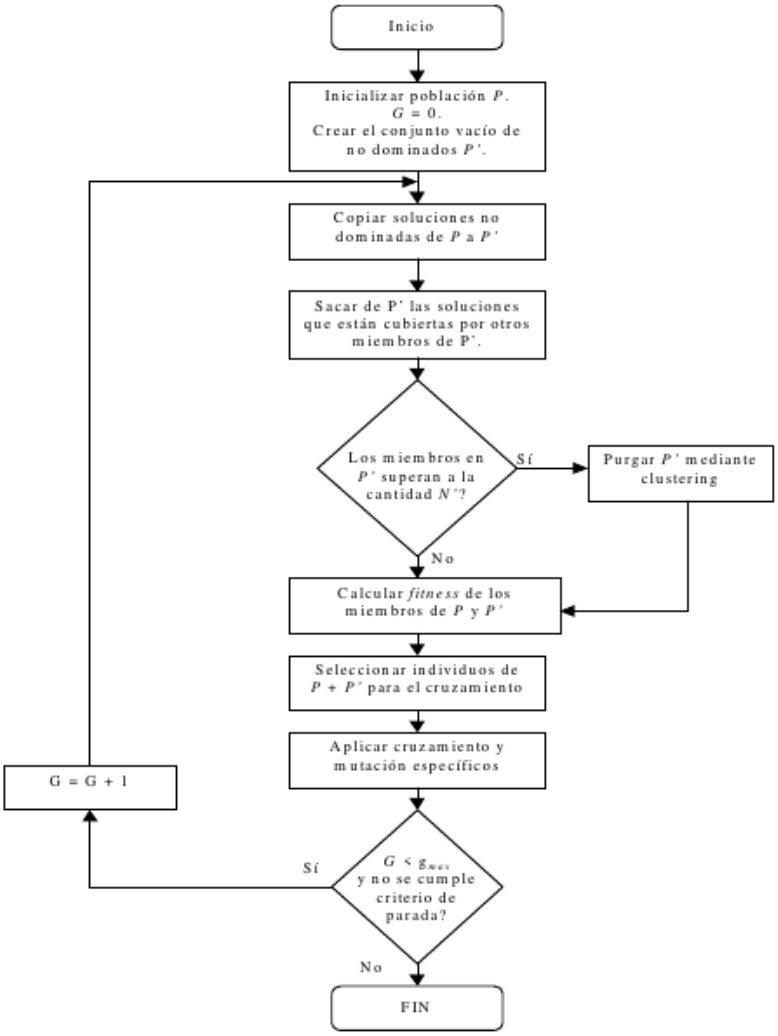


Figura 24: Diagrama de flujo del algoritmo SPEA

También conocido como *Strength Pareto Evolutionary Algorithm* (Zitzler y Thiele, 1998). Entre sus características más significativas, destacan:

- Dominancia de conteo y rango.
- Elitismo mediante un archivo de individuos no dominados.
- Valor de *ranking* de individuos similar al MOGA.
- Realiza *clustering* (*average linkage method*) para mantener la diversidad y así reducir el número de soluciones no dominadas guardadas sin destruir las características del frente de Pareto óptimo.

Las etapas del algoritmo son:

1. Inicializar el archivo externo $P^e = \emptyset$ y la población inicial P.
2. Copiar las soluciones no dominadas de la población actual al archivo. Se quitan en P^e aquellas soluciones dominadas por otras de este conjunto, más los duplicados del conjunto. Si el archivo sobrepasa su capacidad, reducirlo mediante *clustering*.
3. Calcula el *fitness* de los individuos de la población actual más los del archivo (denominado P'). El *fitness* se obtiene calculando el *strength* de la solución i en P^e: $S_i = n_i \ / \ (N+1)$ donde n_i es la cantidad de individuos dominados por i en el archivo. Por lo tanto, el *fitness*

del cromosoma j en P se calcula de la siguiente forma: $f_j = 1 + \sum_{i \in P^e \wedge \text{i domina j}} S_i$ (considerando un criterio de minimización).

4. Seleccionar N individuos a partir de P' (la población más las soluciones externas) mediante torneo binario. Dos individuos son aleatoriamente escogidos. Respecto a su función *fitness*, el mejor es escogido para aparearse.

5. Aplicar cruce y mutación.

6. Si la condición de parada no se cumple ir al segundo paso. En caso contrario devuelve los miembros del archivo como solución.

El algoritmo de *clustering* consiste en realizar los siguientes pasos: primero cada individuo se asocia a un clúster y en caso de que el archivo no esté lleno, elegir un individuo que tenga la mínima distancia media de cada clúster; el segundo paso consiste realizar un par de clústers para calcular la distancia media entre todos los individuos del par (la fórmula para calcular esa distancia sería $D_{ij} = 1/|C_i| * |C_j| \sum_{i \in C_i, j \in C_j} d(i,j)$); el tercer paso consistiría unir el par de clústers cuya distancia entre ellos sea mínima; el cuarto paso consisten en que si el número de clústers es menor o igual que la capacidad del archivo, entonces ir al siguiente paso, en otro caso ir al segundo paso; el quinto paso consistiría en elegir un individuo de cada clúster cuya distancia media al resto de individuos del clúster sea la menor. Esto último se realiza iterativamente combinando clústers adyacentes hasta que el número requerido de individuos es alcanzado. El algoritmo de *clustering* se puede aplicar tanto en el espacio de decisión como en el espacio objetivo (como sugieren los autores).

Se sugiere que el ratio entre poblaciones sea de 1:4 (ejemplo: $|P^e| = 20$, $|P| = 80$). Como desventajas de este algoritmo nos encontramos que el coste computacional puede llegar a ser muy elevado en el paso del *clustering*. Los resultados del SPEA suelen ser mejores que el NSGA con un algoritmo secuencial, mientras que los resultados del NSGA con elitismo son similares al SPEA. Entre sus ventajas se encuentra que cumple con el teorema de convergencia de algoritmos evolutivos.

La revisión de este algoritmo se denomina SPEA2 (para corregir las debilidades de la primera versión), cuyas diferencias respecto a su antecesor son las siguientes:

- Asignación del *fitness*: En el SPEA, para un individuo se toma en cuenta la cantidad de individuos que domina y que lo dominan. Esto significa que los individuos que son dominados por el mismos miembros del archivo tienen valores de *fitness* idénticos, de tal forma que si el archivo tiene solo un único individuo, toda la población tiene el mismo ranqueo, independientemente de si es dominado por algún miembro de la población o no. En consecuencia, la presión de selección decrece sustancialmente y se comporta como un algoritmo de búsqueda aleatorio. Con esta nueva versión, incorpora una estrategia de asignación *fine-grained* para que un individuo toma en cuenta la cantidad de individuos que domina y que lo dominan.

- Estimación de densidad: En el SPEA, si algunos individuos de la actual generación son indiferentes y ninguno domina al otro, ninguna o muy poca información puede ser obtenida por la relación de dominancia. En esta situación, que se suele producir con más de dos objetivos, la información de densidad tendría que ser usada para realizar una búsqueda guiada más efectiva. El *clustering* hace uso de esta información, pero solo considera la del archivo y no la de la población. Con esta nueva versión, emplea una técnica de estimación de densidad de vecinos más próximos para dirigir mejor la búsqueda.

- Truncación del archivo: Aunque el SPEA contiene una técnica de *clustering* que para reducir el conjunto no dominado sin destruir sus características, puede perder soluciones externas, y estas soluciones deberían mantenerse en el archivo para obtener una buena dispersión de las soluciones no dominadas. Posee un método mejorado de truncado para garantizar la preservación de los individuos en los límites.

El tamaño de archivo es fijo, por lo que se necesita una técnica de *clustering*, realizada en el paso de selección de entorno. Otra diferencia con el SPEA es que solo los miembros del archivo participan en el proceso de selección de apareamiento.

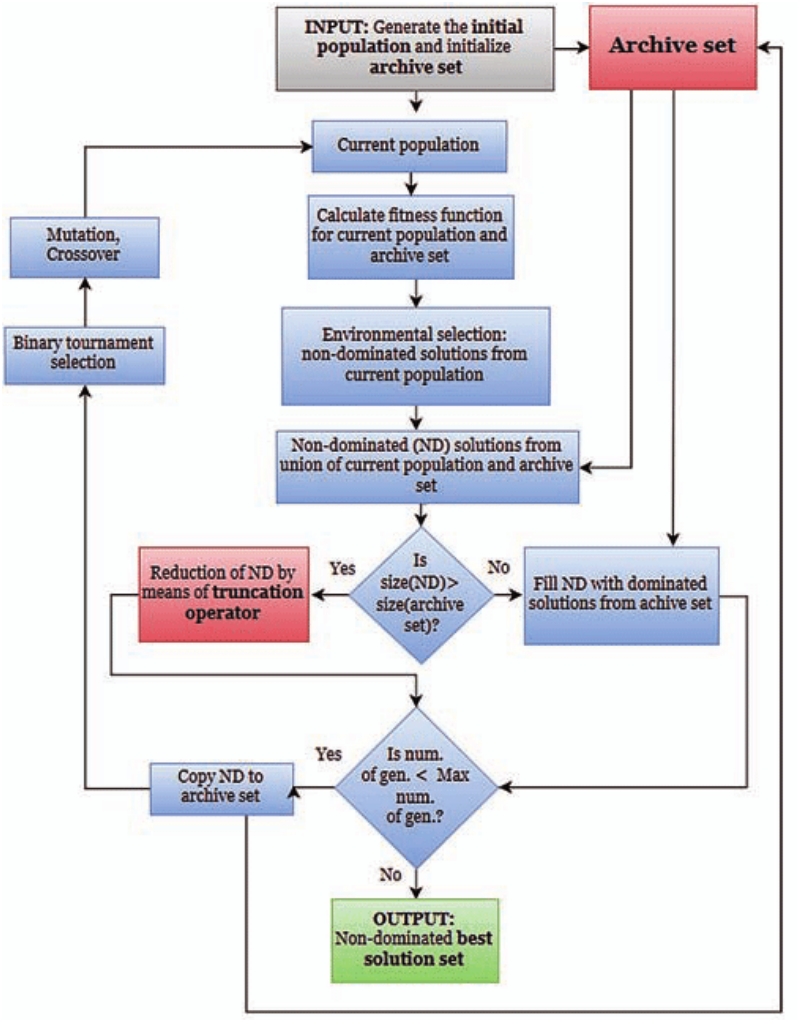


Figura 25: Diagrama de flujo del algoritmo SPEA2

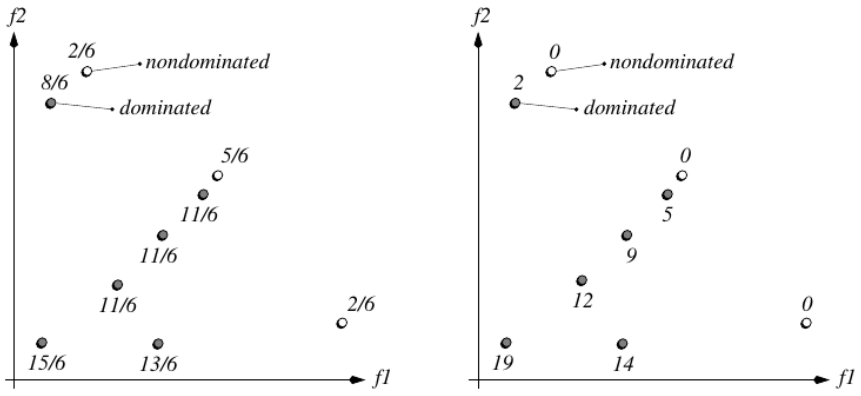


Figura 26: Comparación de asignamiento de fitness en SPEA y SPEA2
La imagen visualiza un problema de maximización con dos objetivos, f_1 y f_2. A la izquierda, los valores de *fitness* de una población son dadas de acuerdo al esquema de SPEA. A la derecha, se corresponden los valores de *fitness* de acuerdo al esquema de SPEA2

Los argumentos son los siguientes:

- N: tamaño de la población
- $\overline{N}$: tamaño del archivo
- T: máximo número de generaciones

La salida es el conjunto no dominado.

1. Inicialización: Genera una población inicial P_0, un archivo $\overline{P}_0 = \emptyset$ y definimos $t = 0$.
2. Asignamiento del *fitness*: Calcula los valores de *fitness* de los individuos en P_t y $\overline{P}_t$.
3. Selección de entorno: Se copian todas las soluciones no dominados en P_t y $\overline{P}_t$ a $\overline{P}_{t+1}$. Si el tamaño de $\overline{P}_{t+1}$ excede $\overline{N}$ entonces se reduce $\overline{P}_{t+1}$ mediante un operador de truncación, de no ser así si el tamaño de $\overline{P}_{t+1}$ es menor que $\overline{N}$ entonces rellenar $\overline{P}_{t+1}$ con individuos dominados de P_t y $\overline{P}_t$.
4. Terminación: Si $t \geq T$ o otro criterio de parada es satisfecho entonces devolver los individuos de $\overline{P}_{t+1}$ y parar.
5. Selección de apareamiento: Se realiza una selección por torneo binario con reemplazamiento en $\overline{P}_{t+1}$ para rellenar la piscina de apareamiento.
6. Variación: Aplicar operadores de recombinación y mutación a la piscina de apareamiento y definir P_{t+1} con el resultado que proporcione. Incrementar el contador $t = t + 1$ y ir al paso 2.

Para evitar la situación de individuos dominados por los miembros del archivo con idéntico valor de *fitness*, en SPEA2 cada individuo tanto los individuos dominados como los dominantes son tomados en cuenta. En detalle, cada individuo i en el archivo $\overline{P}_t$ y la población P_t es asignado por un valor de fuerza $S(i)$, representando el número de soluciones que domina:

$$\boxed{S(i) = \left|\{j | j \in P_t + \overline{P}_t \wedge i \succ j\}\right|} \tag{26}$$

donde $|\cdot|$ representa la cardinalidad del conjunto, $+$ es la unión multiconjunto y el símbolo $\succ$ se corresponde a la relación Pareto dominancia. Con los valores de S, el *fitness* bruto $R(i)$ de un individuo i es calculado:

$$\boxed{R(i) = \sum_{j \in P_t + \overline{P}_t, j \succ i} S(j)} \tag{27}$$

El *fitness* en bruto es determinado por las fuerzas de sus dominadores que se encuentran tanto en el archivo como en la población, al contrario que el SPEA donde solo los miembros del archivo son considerados. Es importante notar que el *fitness* debe ser minimizado, por ejemplo, $R(i) = 0$ se corresponde a un individuo no dominado, donde un alto valor de $R(i)$ significa que i es dominado por muchas soluciones.

Aunque el *fitness* en bruto provee un ordenado de nichos basado en el concepto de Pareto dominancia, puede fallar cuando muchos individuos no dominan al resto. Por lo tanto, una información adicional de densidad es incorporada para discriminar entre individuos que tienen valores de *fitness* en bruto idénticos. La técnica de estimación de densidad usado en SPEA2 es una adaptación del método de k-vecinos más cercanos (Silverman, 1986), donde la densidad en cualquier punto es una función (decreciente) de la distancia a cada k punto más cercano. Aquí simplemente se toma la inversa de la distancia del k vecino más cercano como estimación de densidad. Para ser más preciso, para cada individuo i, las distancias (en el espacio objetivo) de todos los individuos j en el archivo y la población son calculados y guardados en una lista. Después del ordenado de la lista en orden creciente, el k elemento da la distancia buscada, denominada como σ_i^k. Como característica común, usamos la k igualdad para la raíz cuadrada del tamaño de la muestra (Silverman, 1986), por tanto, $k = \sqrt{N + \overline{N}}$. Por lo tanto, la densidad $D(i)$ correspondiente a i es definida como

$$\boxed{D(i) = \frac{1}{\alpha_i^k + 2}} \tag{28}$$

En el denominador, el dos es añadido para asegurar que este valor sea mayor que cero y que $D(i) < 1$. Finalmente, añadiendo $D(i)$ al valor del *fitness* en bruto $R(i)$ de un individuo i conduce al *fitness* $F(i)$:

$$\boxed{F(i) = R(i) + D(i)} \tag{29}$$

El tiempo de ejecución del asignamiento de *fitness* es dominado por el estimador de densidad $\mathcal{O}(M^2 \log M)$, mientras que el cálculo de S y R es de complejidad $\mathcal{O}(M^2)$, donde $M = N + \overline{N}$.

La actualización del archivo (paso 3 en el algoritmo) en SPEA2 difiere del SPEA en dos aspectos: el número de individuos que contiene el archivo es constante siempre y el método de truncación previene que las soluciones en los límites sean eliminadas.
La definición matemática que se realiza durante la selección de entorno se describe:

$$\boxed{\overline{P}_{t+1} = \{i \mid i \in P_t + \overline{P} \wedge F(i) < 1|} \tag{30}$$

Si el frente no dominado encaja exactamente en el archivo ($|\overline{P}_{t+1}| = \overline{N}$) la selección de entorno es completada. En otro caso, pueden darse dos situaciones. Si el archivo es demasiado pequeño ($|\overline{P}_{t+1}| < \overline{N}$) o demasiado largo ($|\overline{P}_{t+1}| > \overline{N}$). En el primer caso, los mejores $\overline{N} - |\overline{P}_{t+1}|$ individuos dominados en el archivo previo y población son copiados al nuevo archivo. Esto puede ser implementado ordenando el conjunto $P_t + \overline{P}_t$ de acuerdo a su valor de *fitness* y copiamos los primeros $\overline{N} - |\overline{P}_{t+1}|$ individuos i con $F(i) \geq 1$ a $\overline{P}_{t+1}$. En el segundo caso, cuando el tamaño del actual conjunto no dominado excede $\overline{N}$, una truncación de archivo es invocada que iterativamente elimina individuos de $\overline{P}_{t+1}$ hasta $|\overline{P}_{t+1}| = \overline{N}$. Aquí, en cada iteración donde el individuo i es elegido para eliminar para cada $i \leq_d j$ para cada $j \in \overline{P}_{t+1}$ con

$$\boxed{\begin{array}{lll} i \leq_d j & :\Leftrightarrow & \forall\ 0 < k < |\overline{P}_{t+1}| : \sigma_i^k = \sigma_j^k \quad \vee \\ & & \exists\ 0 < k < |\overline{P}_{t+1}| : [(\forall\ 0 < l < k : \sigma_i^l = \sigma_j^l) \wedge \sigma_i^k < \sigma_j^l] \end{array}} \tag{31}$$

donde σ_i^k denota la distancia de i a su k vecino más cercano en $\overline{P}_{t+1}$, es decir, el individuo que tiene la mínima distancia respecto a otro es escogido en cada fase; si hay varios individuos con mínima distancia el vínculo es roto por considerar la segunda mínima distancias y las siguientes.
Aunque la peor complejidad temporal para el operador de truncación es $\mathcal{O}(M^3)(M = N + \overline{N})^2$, la media de complejidad debería ser pequeña $\mathcal{O}(M^2 \log M)$ ya que los individuos suelen diferir con considerar el segundo o tercero vecino más cercano, y por eso el ordenado de las distancias gobiernan totalmente la complejidad.

5.10.4 PAES

La finalidad del PAES es la de poder representar de la forma más simple posible un algoritmo no trivial capaz de generar diversas soluciones en el conjunto óptimo de Pareto. El algoritmo usa una estrategia evolutiva (1 + 1), utilizando búsqueda local de una población además de un archivo. Fue desarrollado con dos objetivos principales en mente:

- El primero fue que el algoritmo fuese estrictamente confinado a la búsqueda local, por ejemplo, debería usar solo un pequeño operador de cambio (mutación) y moverlo desde la actual solución a un vecino cercano. Esto es lo que diferencia de otras aproximaciones como el NPGA, NSGA o VEGA, que mantienen una población de soluciones cuando se realizan la selección y cría del algoritmo genético.
- El segundo objetivo fue que el algoritmo debía ser un verdadero optimizador de Pareto, tratando todas las soluciones no dominadas con un mismo valor. Esto es difícil ya que en la mayoría de casos al comparar dos soluciones ninguna domina a la otra. Este problema es solventado manteniendo un archivo de soluciones no dominadas encontradas previamente. Este archivo es usado para estimar la verdadera dominancia entre dos soluciones.

Se puede ver PAES como tres partes:

- La función de aceptación de las soluciones candidatas.
- El generador de candidatos a soluciones.
- El archivo de soluciones no dominadas.

Visto desde este punto de vista, PAES representa la aproximación no trivial más simple de un algoritmo multiobjetivo de búsqueda local. El generador de candidatos a soluciones es llevado a cabo con una simple mutación aleatoria de escalada; mantiene la actual solución (única). Como el

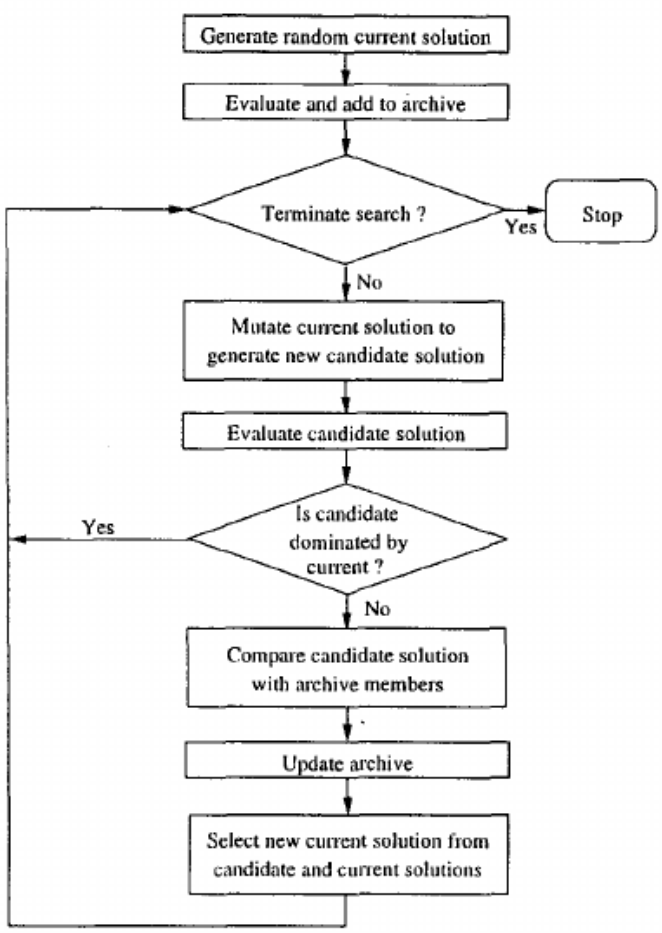


Figura 27: Diagrama de flujo del algoritmo PAES

tamaño del archivo es limitada, el diseño de la función de aceptación para escoger entre la mutación o la actual solución es una aproximación diseñada de Horn et al.

El algoritmo empieza con la inicialización de un único cromosoma (la solución actual) que es evaluada usando la función de coste multiobjetivo. Una copia es creada y el operador de mutación es aplicado a la copia. Esta copia mutada es evaluada y forma el nuevo candidato a solución. La solución actual y la solución candidata deben ser comparadas. Si una domina a la otra, la aceptación es simple, pero en caso de que ninguna domine a la otra, se utiliza como referencia un archivo de soluciones no dominadas calculado anteriormente. Si la comparación con el archivo de población falla en favor de una solución respecto a la otra, el desempate se realiza escogiendo la solución que resida en la región menos concurrida del espacio.

El archivo sirve para dos propósitos. El primero, guarda y actualiza todas las soluciones no dominantes (sujeto a varios criterios) generadas. Segundo, durante la ejecución, es usado como ayuda para precisar la selección entre la actual solución y la solución candidata hacia el actual frente de soluciones no dominadas. Sin este proceso, el algoritmo no es capaz de diferenciar entre buenas y malas decisiones con el resultado de que el algoritmo deambula sin rumbo sobre el espacio de búsqueda.

El proceso de archivado es similar que el propuesto por G.T. Parks y I. Miller (1998), pero no es usado como piscina para la selección. El archivo tiene un tamaño máximo, $refpop$, que es dado por el usuario para reflejar el número requerido de soluciones finales deseadas. Cada candidato a solución generada que no es dominado por sus padres (la solución actual) es comparada con cada miembro del conjunto de comparación. Candidatos que dominan todo el conjunto de comparación son siempre aceptados y archivados. Candidatos que son dominados por el conjunto de comparación son siempre rechazados, mientras aquellos que no son dominados son aceptados y/o archivados

basado en el grado de cruzamiento en su localización en la malla. Este archivado y aceptado lógico está explicado explícitamente en la siguiente figura. Para mantener de la pista de los grados de

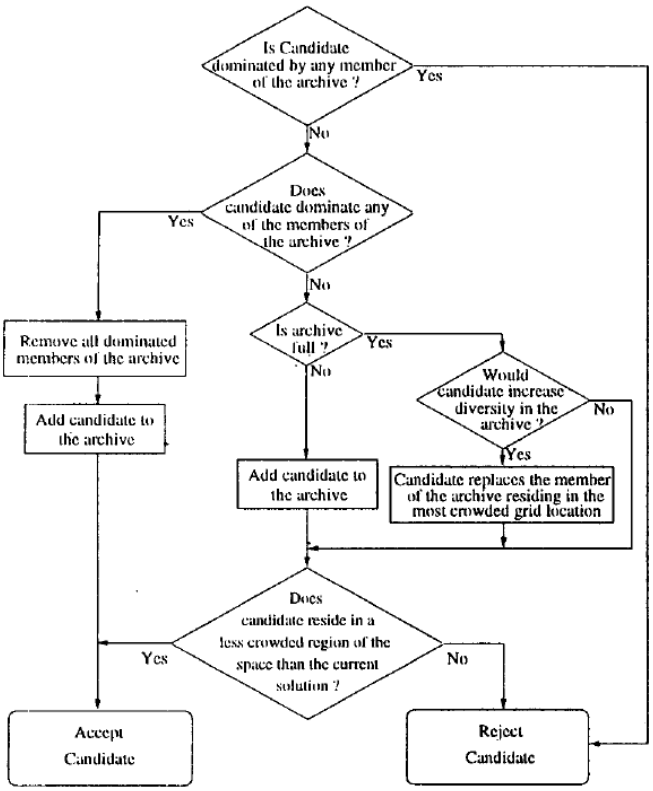


Figura 28: Lógica de archivado y aceptación del algoritmo PAES

crowding en diferentes regiones del espacio de solución, un *grid* d-dimensional es usado, donde d es el número de objetivos en el problema. Cuando cada solución es generada, se localiza en el *grid* usando una subdivisión recursiva y usando un árbol de codificación. Un mapa del *grid* es también mantenido, indica cada localización del *grid* y cuantas soluciones del archivo residen en ese grid. Cuando una solución candidata esta en posición de unirse al archivo lleno, lo reemplaza por una solución del archivo que tenga la mayor cantidad de miembros en su localización del *grid*, siempre y cuando el conteo de la localización del grid del candidato es menor. Este sistema también es usado para seleccionar entre la actual y las soluciones candidatas cuando el candidato no es dominado ni domina ningún miembro del archivo. En este caso, la solución con menor conteo en el *grid* es seleccionado.

PAES solo recibe dos parámetros. Estos son el tamaño del archivo, $refpop$ y el número de subdivisiones del espacio en la malla usado para promover la diversidad. Con $refpop$ por debajo de 50, el rendimiento del algoritmo se degrada. Se recomienda que se ponga a 100, y a estos niveles, el archivo raramente se llena durante unas 20000 evaluaciones. Con el conjunto $refpop$ por debajo de 50, el rendimiento del algoritmo empieza a degradarse.

5.10.5 MOEA/D

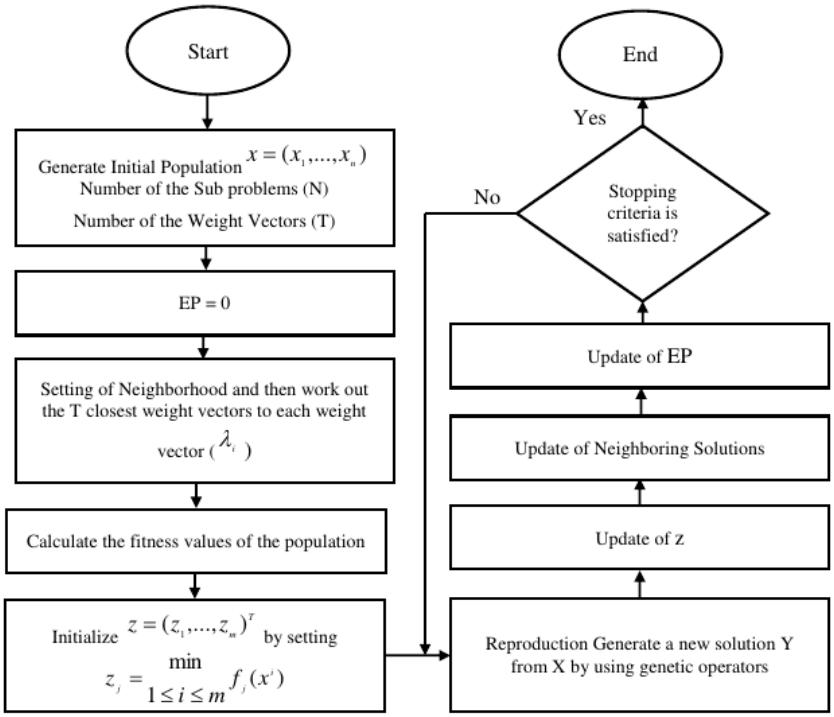
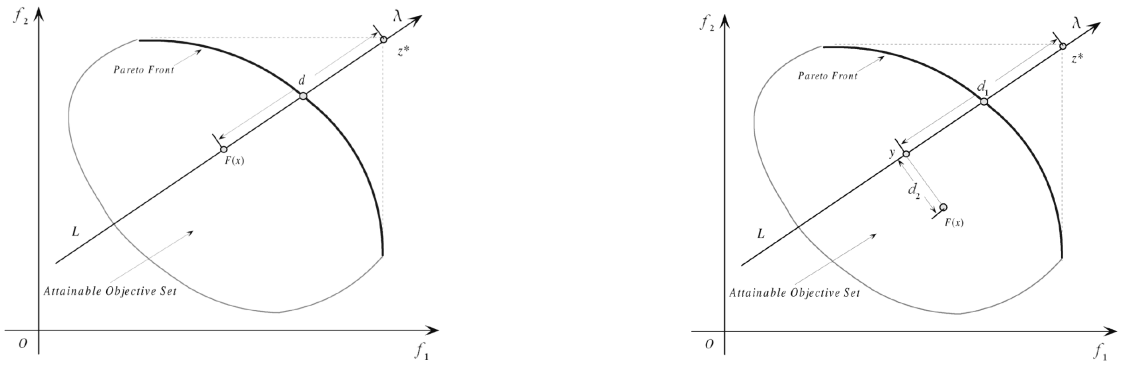


Figura 29: Diagrama de flujo del algoritmo MOEA/D

Se trata de un algoritmo genético multiobjetivo basado en la descomposición explicado por Qingfu Zhang (2007). Hay varias aproximaciones, pero en esta explicación seguiremos la aproximación de Tchebycheff. La descomposición suele ser bastante trivial. Antes de explicar el algoritmo, hay que comentar las diversas aproximaciones más conocidas para descomponer un problema de optimización multiobjetivo en un problema de optimización escalar:

- Suma de pesos, ε-constante y la aproximación de Tchebycheff (mencionados anteriormente en los métodos clásicos de los MOP). Este último entraría dentro de la función de la distancias.
- Aproximación de la intersección límite: Recientes métodos de descomposición de MOP como el método de intersección del límite normal o el de la constante normal normalizada forman parte del conjunto de esta aproximación. Han estado diseñados para un MOP continuo. Bajo algunas condiciones reculares, el frente de Pareto de un MOP continuo es parte del mayor límite de la derecha. Geométricamente, estas aproximación tienen como objetivo encontrar los puntos de intersección en el mayor límite y el conjunto de líneas. Si estas líneas son distribuidas, podemos esperar que los puntos resultantes de la intersección aportan una buena aproximación al frente de Pareto. Estas aproximaciones pueden trabajar con frentes de Pareto no cóncavos. La ventaja de estas aproximaciones sobre la aproximación de Tchebycheff son que en caso de tener más de dos objetivos, si los dos usan el mismo conjunto de vectores de pesos distribuido, las soluciones resultantes del la aproximación de la intersección límite suelen estar más uniformemente distribuidas. Como desventaja tiene que un factor de penalización alto o bajo repercute bastante en el rendimiento.



(a) Aproximación

(b) Penalización

Figura 30: Intersección límite

Teniendo $\lambda^1, ..., \lambda^N$ como un conjunto de vectores de pesos esparcidos y z^* como punto de referencia. Este puede ser descompuesto en N subproblemas de optimización escalar usando cualquier aproximación comentada en el párrafo anterior. MOEA/D minimiza todas esas N funciones objetivo simultáneamente en una pasada. En el MOEA/D, un vector de pesos de vecindad λ^i es definido como un conjunto de los vectores de pesos más cercanos en $\{\lambda^1, ..., \lambda^N\}$. La vecindad del i-ésimo problema consiste en todos los subproblemas con los vectores de pesos de la vecindad de λ^i. La población es compuesta de la mejor solución encontrada para cada subproblema. Solo las actuales soluciones de la vecindad de los subproblemas son explotadas para optimizar un subproblema en MOEA/D.

A cada generación t, MOEA/D, con la aproximación de Tchebycheff, mantiene:

- una población de N puntos $x^1, ..., x^N \in \Omega$ donde x^i es la actual solución del i-ésimo subproblema.
- $FV^1, ..., FV^N$, donde FV^i es el F-valor de x^i, por ejemplo, $FV^i = F(x^i)$ para cada $i = 1, ..., N$
- $z = (z_1, ..., z_m)^T$, donde z_i es el mejor valor encontrado para el objetivo f_i
- una población externa, que es usada para almacenar las soluciones no dominadas durante la búsqueda

El algoritmos funciona de la siguiente manera. Como entrada recibe:

- el MOP
- criterio de parada
- N: el número de subproblemas considerados
- una extensión uniforme de N vectores de pesos: $\lambda^1, ..., \lambda^N$
- T: el número de vectores de pesos en la vecindad de cada vector de peso.

Como salida da el frente de Pareto EP. El algoritmo se divide en tres pasos: inicialización, actualización y criterio de parada. Empezaremos por la inicialización:

1. $EP = \emptyset$
2. Calcular las distancias euclídeas entre cualquier dos vectores de pesos y entonces calculamos los T vectores de pesos más cercanos a cada vector de peso. Para cada $i = 1, ..., N$, definir $B(i) = \{i_1, ..., i_T\}$, donde $\lambda^{i1}, ..., \lambda^{iT}$ son los T vectores de pesos más cercanos a λ^i
3. Generar una población inicial $x^1, ..., x^N$ aleatoriamente o por un método específico del problema. Definir $FV^i = F(x^i)$
4. Inicializar $z = (z_1, ..., z_m)^T$ mediante un método específico del problema.

El segundo paso, el de actualización, procede como dice a continuación, para cada $i = 1, ..., N$ hacer:

1. Reproducción: Seleccionamos aleatoriamente dos índices k, l de $B(i)$, y entonces generamos una nueva solución y de x^k y x^l usando operadores genéticos.
2. Mejora: Aplicar un método de reparación para el problema o una heurística en y para producir y'.
3. Actualización de z: Para cada $j = 1, ..., m$, si $z_j < f_j(y')$, entonces definir $z_j = f_j(y')$.
4. Actualizar las soluciones vecinas: Para cada índice $j \in B(i)$, si $g^{te}(y') \leq g^{te}(x^j)$, entonces $x^j = y'$ y $FV^j = F(y')$.
5. Actualización de EP: Eliminar de EP todos los vectores dominados por $F(y')$ y añadir $F(y')$ a EP si no hay vectores en EP que dominan $F(y')$.

Por último está el criterio de parada, que si es satisfecho, entonces se detiene y da como salida EP, en otro caso ir al paso de actualización.

En la inicialización, $B(i)$ contiene los índices del T (tamaño de la vecindad) vectores más cercanos de λ^i. Usamos una distancia euclídea para medir la proximidad entre dos vectores de pesos. Por lo tanto, el vector más cercano a λ^i es él mismo, y entonces $i \in B(i)$. Si $j \in B(i)$, el j-ésimo subproblema puede ser considerado como vecino del i-ésimo subproblema.

En el i-ésimo paso en el bucle en el paso 2, los T subproblemas vecinos del i-ésimo subproblema son considerados. Desde que x^k y x^l en el paso 2.1 son las mejores soluciones actuales en la vecindad del i-ésimo subproblema, su descendencia y debería ser una buena solución para el i-ésimo subproblema. En el paso 2.2, un problema específico de heurística es usado para reparar y en el caso que y invalidase cualquier constante, y/o optimizase el i-ésimo g^{te} (función a optimizar). Entonces, la solución y' resultante es factible y muy propensa a tener un valor de función bajo para los vecinos del i-ésimo subproblema. El paso 2.4 considera todas las vecindades del subproblema i-ésimo, y reemplaza x^j con y' si y' es mejor que x^j considerando el j-ésimo subproblema. FV^j es necesario para computar el valor de g^{te} en el paso 2.4.

Desde que consume mucho tiempo el encontrar el punto de referencia exacto z^*, usaremos z, que es inicializado en el paso 1.4 mediante un método y actualizado en el paso 2.3, como sustituto de z^* en g^{te}. La población externa, inicializada en el paso 1.1, es actualizada por la nueva solución generada y' en el paso 2.5. En el caso de que el criterio de la función sea minimizar $F(x)$, la inequalidad en el paso 2.3 debería ser revertida.

Por último, hay que tener en cuenta que como disponemos de recursos limitados, optimizar todas las posibles funciones de agregación no es muy práctico. A diferencia de por ejemplo, NSGA-II y SPEA2 que usan distancias de *crowding* entre las soluciones para mantener la diversidad, no siempre es fácil generar una distribución uniforme de los Pareto óptimo con esta última técnica, sin embargo, como MOEA/D descompone el MOP en varios subproblemas de optimización escalar, en caso de que se escoja un buen método de descomposición y vectores de peso, MOEA/D tiene una buena probabilidad de producir una distribución uniforme de las soluciones Pareto si se optimizan todos estos subproblemas correctamente. Finalmente, si solo tenemos dos subproblemas vecinos, en caso de que T sea pequeño, dos soluciones elegidas para los operadores genéticos suelen generar descendencia muy parecida al estar sus padres cerca, lo que limita la habilidad de explorar nuevas áreas en el espacio de búsqueda. Por otra parte, si T es muy grande, se debilita la habilidad de explotación, además de aumentar el computo del paso 2.4.

5.10.6 Otros

Hay muchas más aproximaciones de algoritmos genéticos multiobjetivo si incluimos modificaciones de estas además de modificaciones para dar más énfasis a la búsqueda local, incluso algoritmos genéticos basado en el óptimo de Pareto que no son parecidos a los más conocidos y que aportan una asignación del *fitness*, selección o otro paso del algoritmo genético como novedad. Un listado no actualizado pero amplio puede ser consultado en [DEB01].

- MOMGA/MOMGA II/MOMGAIII: *Multi-Objective Messy GA*.
- DPGA: *Distance-based Pareto Genetic Algorithm* (Osyczka y Kundu, 1995).
- cNSGA II: *controlled Non-dominated Sorting Genetic Algorithms* (versión del NSGA).
- AENSGA (versión del NSGA).
- NCGA: *Neighborhood Cultivation Genetic Algorithm.*
- iPESA II.
- M-PAES.
- MOGLS.
- cMOGA: *cellular Multi-Objective Genetic Algorithm* (Murata et al.) (parecido al MOEA/D).
- RMOGA: *Robust Multi-Objective Genetic Algorithm.*

5.11 3ª generación

El área que requiere de más investigación en MOEA es la fundamentación teórica. La mayor parte de la investigación actual se concentra en demostrar la convergencia en la optimización multiobjetivo en los algoritmos evolutivos. Además de esta línea principal, hay algunas otras que aún se encuentran en discusión, tales como:

- Estudiar más a fondo la estructura de las hipersuperficies de aptitud para los problemas de optimización multiobjetivo.
- Responder a la siguiente cuestión: ¿Cuál es la dificultad en la optimización multiobjetivos para un algoritmo evolutivo?
- Un desarrollo formal para analizar y demostrar la convergencia en algoritmos evolutivos multiobjetivo paralelos.
- Buscar el límite teórico para el *ranking* de Pareto asumiendo poblaciones de tamaño finito.
- Hacer un análisis en tiempo real del rendimiento de un algoritmo evolutivo multiobjetivo.
- Tener definiciones aceptadas por la comunidad sobre robustez, convergencia y diversidad, entre otras en el contexto de optimización evolutiva multiobjetivo.
- Incorporar preferencias: Muchas de las actuales técnicas usadas en la toma de decisiones multicriterio que se desarrolla en el área de investigación operativa aún no son bien aplicadas en optimización evolutiva multiobjetivo. Esta incorporación es muy importante para aplicaciones reales ya que el usuario en realidad solo necesitará una sola solución Pareto óptima en vez del conjunto entero como ocurre en los algoritmos evolutivos multiobjetivo.
- Optimización de funciones dinámicas: Es el siguiente paso, luego de atacar funciones estáticas con varios objetivos. En este caso, las fronteras de Pareto se mueven en el tiempo debido a la existencia de variables aleatorias o dinámicas.
- Tratamiento de problemas de solución de problemas multiobjetivo con espacios de búsqueda altamente restringidos. Son problemas bastante comunes y es necesario desarrollar técnicas nuevas para el manejo de este tipo de restricciones que puedan lidiar con este tipo de problemas.
- Es necesario tener más algoritmos y también modelos formales que comprueben la convergencia y más aplicaciones reales que usen paralelismo.

Los algoritmos genéticos multiobjetivo tuvo su apogeo en la primera década del 2000 hasta que se empezaron a estudiar el resto de algoritmos evolutivos y técnicas no basadas en algoritmos evolutivos, sin embargo, estos no sustituyeron a los algoritmos genéticos multiobjetivo sino que surgieron como alternativas, así que la proliferación de los algoritmos genéticos ha disminuido, aunque esto no ha impedido que se hayan ido refinando con el paso del tiempo. Recientemente se

han lanzado versiones tanto del NSGA-III como del SPEA III [RUD15] (este no por sus antiguos desarrolladores) actualizadas, pero sin ninguna mejora añadida, simplemente se han implementado métodos para generalizar los algoritmos, es decir, que se puedan aplicar de forma más sencilla a un mayor número de problemas.

5.11.1 NSGA-III

El NSGA-III fue propuesto por Deb and Jain (2014) cambiando algunos mecanismos de selección. Está basado en los siguientes pasos:

begin
  Calcular el número de puntos de referencia (H) a colocar en el hiperplano;
  Generar la población inicial aleatoriamente teniendo en cuenta los recursos de asignamiento a constantes (cromosomas POP);
  Realizar el ordenado de la población no dominada;
  for *i=1* to criterio de parada **do**
    Seleccionar dos padres $P1$ y $P2$ usando el método de torneo;
    Aplicar el cruce entre $P1$ y $P2$ con una probabilidad P_c;
    Realizar el ordenado de la población no dominada;
    Normalizar los miembros de la población;
    Asociar los miembros de la población con sus puntos de referencia Aplicar la preservación de contador de nicho;
    Mantener las soluciones obtenidas del nicho para la siguiente generación;
  end
end

Algoritmo 9: NSGA-III

Hay que definir un conjunto de puntos de referencia en el hiperplano para asegurar la diversidad de las soluciones obtenidas. Diferentes puntos son colocados en el hiperplano normalizado que tienen la misma orientación en todos los ejes. El número de puntos de referencia (H) es definido por:

$$\boxed{H = \begin{pmatrix} C+g-1 \\ g \end{pmatrix}} \tag{32}$$

donde C es el número de la función objetivo, g es el número de divisiones a considerar en cada eje objetivo (para 3 objetivos y 4 divisiones, tendremos 15 puntos de referencia). Los puntos de referencia son colocados en el hiperplano y las soluciones serán descritas por el frente de Pareto, entonces las soluciones serán asociadas con los creados puntos de referencia.

Tenemos que determinar el punto ideal de la población actual, así que tenemos que identificar el mínimo valor de cada función objetivo ($OF_i^{min}, i = 1, 2, ..., C$). Entonces, cada función objetivo será movida sustrayendo $z_i^{min} = (OF_1^{min}, OF_2^{min}, ..., OF_C^{min})$ al objetivo f_i. Continuamos con los pasos propuestos por Deb y Jain (2014) para generar un hiperplano (resolviendo los frentes de Pareto donde las soluciones objetivo son de diferente escala).

Después de normalizar cada función objetivo, es necesario asociar cada miembro de la población con una referencia. Definimos una línea de referencia para cada punto juntando el punto de referencia con el punto de origen. Entonces, determinamos la distancia perpendicular entre cada miembro de la población y cada línea de referencia. Finalmente, el punto de referencia que tenga la línea de referencia más cerca de un individuo es asociado a este miembro de la población.

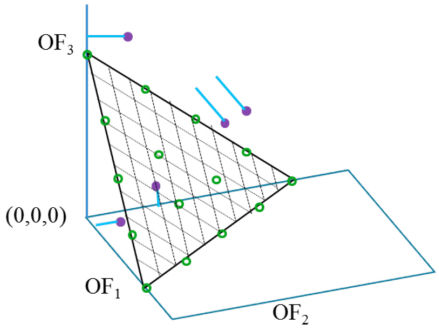


Figura 31: Asociación de soluciones con puntos de referencia

Un punto de referencia puede ser asociado a una o varias soluciones, pero podemos mantener la solución que está más cerca del punto (distancia perpendicular desde la línea de referencia, Deb y Jain (2014)).

La generación de hijos son hechos aplicando los operadores genéticos usados en NSGA-II. Deb y Jain (2014) sugieren fijar el tamaño de la población y que sea parecido al número de puntos de referencia (H) para dar la misma importancia a todos los miembros de la población.

Otro algoritmo basado en el NSGA-III que sirve tanto para problemas monobjetivo como multiobjetivo es el U-NSGA-III.

5.12 Paralelismo

Si bien la paralelización de algoritmos evolutivos han estado presentes desde prácticamente sus inicios, pocas veces se han adaptado a los algoritmos genéticos multiobjetivo. Estos algoritmos son conocidos como *parallel Multiobjective Evolutionary Algorithms* (pMOEA). La utilización de pMOEA une las características propias de los MOEA con las ventajas de la computación paralela además de intentar explorar un espacio mayor de posibles soluciones que los algoritmos genéticos monobjetivo. Los modelos de paralelización de algoritmos genéticos más importantes son:

- Difusión.
- Islas.
- Maestro-esclavo.

En el modelo de islas los diferentes algoritmos genéticos multiobjetivo trabajan en cada isla, pudiendo tener distintos parámetros o estructuras que se deben definir convenientemente, lo que implica que las distintas poblaciones (la población total se divide entre las islas denominándose islas) están explorando en regiones distintas del espacio de búsqueda. El modelo de islas requiere la selección de una política de migración a otra isla mediante algún criterio que señale:

- El modo en que los individuos pueden migrar.
- Número de individuos que van a migrar.
- Frecuencia de migración: Se han explorado varias opciones como: enviar individuos transcurrido un número de generaciones, enviar individuos de forma probabilística, enviar individuos de forma adaptativa, entre otras. En el primer caso, se determina el número de generaciones que deben transcurrir para que se produzca el envío. En el segundo se proporciona como parámetro del algoritmo la probabilidad que un envío ocurra en una determinada generación. La tercera opción se basa en modificar la frecuencia de migración conforme el proceso de evolución ocurre.
- Lugar de dónde se van a escoger los elementos que tienen que migrar.
- El reemplazo de los individuos de una población cuando llegan individuos migrantes de otras poblaciones.

El modelo de islas puede se suele aplicar a sistemas paralelos de memoria distribuida. En esta aproximación son los mejores individuos de una población los que migran cuando se de el caso y hay que especificar como parámetro el número máximo de individuos a enviar.

En la presente aproximación, tanto la recepción como el envío de soluciones se maneja de forma asíncrona, permitiendo que la ejecución de los MOEA continue sin esperas después de recibir o enviar soluciones. Las ventajas que tiene el envío como recepción asíncrona son:

- Correcto tratamiento de pérdida de datos y elementos a procesar.
- Reducción en el tiempo de espera.

Para el reemplazo de individuos de la población por individuos migrantes se puede realizar mediante el reemplazo aleatorio de elementos dominados en los elementos recibidos.

También nos podemos encontrar con pMOEA con topología de migración descentralizada, los cuales forman un equipo de trabajo compuesto por dos tipos diferentes de procesos:

- Un proceso colector.
- p procesos

Durante la ejecución del algoritmo, cada proceso envía a todos los procesos del equipo de trabajo un número determinado de los mejores individuos obtenidos. El colector almacena los individuos recibidos y elimina los individuos dominados y repetidos. En caso de que las soluciones del colector supere su capacidad de almacenamiento se debe proceder a reducir el número de soluciones del colector utilizando, por ejemplo, el procedimiento de *clustering* del SPEA. Cuando finalizan todos los MOEA, se eliminan los elementos dominados en el colector y se devuelven como resultado. Este procedimiento ha sido probado satisfactoriamente en los MOEA representativos de la primera generación (MOGA, NPGA y NSGA) y la segunda generación (SPEA, NSGA-II y CNSGA-II). Para ver ejemplos de algoritmos genéticos multiobjetivo paralelizados mediante el modelo maestro-esclavo como el NSGA o el SPEA, se remite a [DUA01].

Data: parámetros usuales del MOEA, p_{mig} (probabilidad migración), n_{mig} (número elementos no dominados a migrar)
begin
  Unirse al equipo de trabajo;
  Generación aleatoria la población inicial P_0;
  $t = 0$;
  repeat
    if *se reciben soluciones de otro proceso* **then**
      P_{recv} = soluciones recibidas;
      Reemplazar elementos de P_t por elementos en P_{recv};
    end
    Generar nueva población P_{t+1};
    $t = t + 1$;
    if *se cumple criterio de envío* **then**
      P_{mig} = elementos a migrar de P_t;
      Enviar P_{mig} a todos los elementos en el equipo de trabajo;
    end
  until *Alcanzar el criterio de parada*;
  Enviar el frente de Pareto al colector con señal de finalización;
  Salir del grupo de trabajo y terminar la ejecución del algoritmo;
end

Algoritmo 10: Procedimiento MOEA paralelo general

5.13 Evaluación

En los algoritmos genéticos multiobjetivo no existe un único criterio que permita decidir si una aproximación al frente de Pareto es mejor que otra debido a que se pueden considerar diversas medidas de calidad, como por ejemplo:

- La cercanía de la aproximación al frente Pareto óptimo.
- El número de soluciones que forman la aproximación al frente de Pareto.
- La distribución de las soluciones en la aproximación al frente de Pareto.

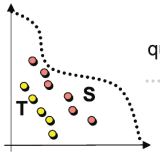
La noción de rendimiento incluye tanto como la calidad del resultado como los recursos computacionales necesitados para generar el resultado. Respecto a este último aspecto, es de práctica común monitorizar ya sea el número de evaluaciones de *fitness* o en general el tiempo de ejecución en un ordenador particular. Al respecto de esto, no hay diferencia respecto a un problema de optimización de un objetivo con un problema de optimización multiobjetivo. En el aspecto de calidad, sin embargo, si hay diferencias. En la optimización de un solo objetivo, podemos definir la calidad en función de la función objetivo: el menor (o mayor) valor, es decir, la mejor solución. Si comparamos dos soluciones en la optimización multiobjetivo, el concepto de Pareto dominancia puede ser usado, aunque, la posibilidad de que dos soluciones sean incomparables, por ejemplo, ninguna solución domina a la otra, complica la situación. Sin embargo, se vuelve aún más complicado cuando comparamos dos conjuntos de soluciones porque algunas soluciones en un conjunto pueden ser dominadas por soluciones en el otro conjunto, mientras otras pueden ser incomparables. Por lo tanto, no esta claro que medida de calidad se puede usar en la aproximación del conjunto de Pareto: la proximidad de las soluciones óptimas en el espacio objetivo, cubrimiento de un amplio rango de diversas soluciones, o otras propiedades? Es difícil definir la medida de calidad apropiada para las aproximaciones del conjunto de Pareto, y como consecuencia se han usado representaciones gráficas para comparar.

A pesar de ello, las medidas de calidad son necesarias para comparar los resultados de los optimizadores multiobjetivo de una manera cuantitativa. Ciertamente, el método más simple de comparación sería el de chequear si un resultado enteramente domina a los otros. La razón, sin embargo, por la cual se han utilizado las medidas de calidad es poder hacer declaraciones más precisas además de eso, que inevitablemente se basan en ciertas suposiciones sobre las preferencias del tomador de decisiones:

- Si un algoritmos es mejor que otro, como puede expresarse cuanto mejor es?
- Si un algoritmo puede decirse que es mejor que otro, en que aspectos podemos decir que es mejor que el último?

Por lo tanto, la clave de la cuestión cuando diseñamos medidas de calidad depende de como de bien resumimos la aproximación del conjunto de Pareto mediante unos números característicos, similares a la estadística, donde podemos encontrarnos la media, la desviación estándar, etc., usadas para describir la distribución de probabilidad de una forma simple. Es inevitable perder información mediante la reducción, y el punto crucial es no perder la información en la que uno está interesado.

Las más populares son medidas de calidad unaria, por ejemplo, la medida asigna a cada aproximación del conjunto de Pareto un número que refleja un cierto aspecto de calidad, y normalmente una combinación de estos es usado. En instancia, la medida de distancia generacional da la distancia media de los vectores objetivo en la aproximación del frente de Pareto bajo consideración del vector objetivo óptimo más cercano, y el hipervolumen considera el volumen del espacio objetivo dominado por la aproximación del frente de Pareto. Normalmente, diferentes medidas de calidad unaria son combinadas, y como resultado de dos aproximaciones del conjunto de Pareto $\boldsymbol{S}, \boldsymbol{T} \subseteq \boldsymbol{X}$ son comparadas comparando las correspondientes medidas de calidad presentes en la siguiente figura. La cuestión es, aunque, que declaraciones pueden ser hechas en base a la información producida por esas medidas de calidad. Es posible concluir con esas medidas de calidad que $\boldsymbol{S}$ es indudablemente mejor que $\boldsymbol{T}$ en el sentido de que $\boldsymbol{S}$, pobremente hablando, domina enteramente a $\boldsymbol{T}$? Esto es crucial en cualquier estudio comparativo.



application of quality measures

	S	T
hypervolume	432.34	420.13
distance	0.3308	0.4532
diversity	0.3637	0.3463
spread	0.3622	0.3601
cardinality	6	5

Figura 32: Ejemplo práctico de medidas de calidad unaria

Zitzler probó las limitaciones teóricas de estas medidas de calidad unaria. En particular, observó que

- no existe ninguna medida de calidad no unaria que pueda ser capaz de indicar que la aproximación del conjunto de Pareto $\boldsymbol{S}$ es mejor que la aproximación del conjunto de Pareto $\boldsymbol{T}$
- el anterior punto se mantiene si se considera un combinación finita de medidas unarias
- la mayoría de medidas que han sido propuestas indican que $\boldsymbol{S}$ es mejor que $\boldsymbol{T}$ y a lo mejor infiere que $\boldsymbol{S}$ no es peor que $\boldsymbol{T}$, es decir, $\boldsymbol{S}$ es mejor o incomparable a $\boldsymbol{T}$
- las medidas unarias son suficientes para detectar que $\boldsymbol{S}$ es mejor que $\boldsymbol{T}$, pero su uso es en general restrictivo
- las medidas de calidad binaria superan las limitaciones de las medidas unarias y, en caso de ser diseñadas adecuadamente, son capaces de indicar si $\boldsymbol{S}$ es mejor que $\boldsymbol{T}$

Esto significa que en genera la calidad de una aproximación del conjunto de Pareto no puede ser completamente descrita por un conjunto finito de distintos criterios como la diversidad o la distancia. Considerando la siguiente combinación de medidas de calidad unaria como la distancia media del frente de Pareto, diversidad de la aproximación del frente de Pareto y el número de vectores en la aproximación del frente de Pareto. En la siguiente figura, la aproximación del conjunto de Pareto $\boldsymbol{S}$ domina todas la soluciones de la aproximación del conjunto de Pareto $\boldsymbol{T}$ (las correspondientes imágenes en el espacio objetivo son visualizadas en la figura). En la parte izquierda, $\boldsymbol{S}$ es juzgado como mejor que $\boldsymbol{T}$ con respecto a todas de las tres medidas de calidad; en la parte derecha, $\boldsymbol{T}$ es asignado mejor en los valores de calidad. Así, esta combinación de medidas de calidad no permite realizar ninguna conclusión acerca si un resultado es mejor que el otro en términos de dominancia de Pareto.

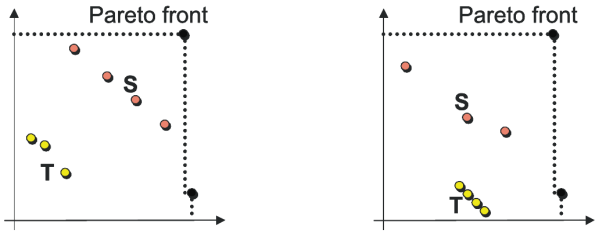


Figura 33: Dos escenarios donde un frente de Pareto domina al otro

Una posibilidad de superar las limitaciones de la medida unario es usar medidas de calidad binaria. En instancia, una medida binaria ε-calidad puede ser definida en base del concepto de la dominancia ε-Pareto.

Si $\boldsymbol{S}$, $\boldsymbol{T} \subseteq \boldsymbol{X}$, entonces la medida binaria de ε $I_\varepsilon(\boldsymbol{S}, \boldsymbol{T})$ es definido como el mínimo $\varepsilon \in \mathbb{R}$ tal que cualquier solución $\boldsymbol{b} \in \boldsymbol{T}$ es ε-dominada por mínimo una solución $\boldsymbol{a} \in \boldsymbol{S}$:

$$\boxed{I_\varepsilon(\boldsymbol{S}, \boldsymbol{T}) = \min\{\varepsilon \in \mathbb{R} \mid \forall \mathbf{b} \in \boldsymbol{T}\ \exists \mathbf{a} \in \boldsymbol{S} : \boldsymbol{a} \succ_\varepsilon \boldsymbol{b}\}} \tag{33}$$

A la vista de la discusión previa, esta medida de calidad posee varias características deseables. Cuando $I_\varepsilon(\boldsymbol{S}, \boldsymbol{T}) < 1$, sabemos que todas las soluciones en $\boldsymbol{T}$ son dominadas por alguna solución en $\boldsymbol{S}$; si $I_\varepsilon(\boldsymbol{S}, \boldsymbol{T}) = 1$ y $I_\varepsilon(\boldsymbol{T}, \boldsymbol{S}) = 1$, entonces $\boldsymbol{S}$ y $\boldsymbol{T}$ representa la misma aproximación del frente de Pareto; si $I_\varepsilon(\boldsymbol{S}, \boldsymbol{T}) > 1$ y $I_\varepsilon(\boldsymbol{T}, \boldsymbol{S}) > 1$, entonces $\boldsymbol{S}$ y $\boldsymbol{T}$ son incomparables, por ejemplo, ambos contienen soluciones no dominadas por el otro conjunto. Además, la medida I_ε representa una extensión natural de la evaluación de aproximación de esquemas en teoría de la computación y da el factor por el cual un resultado es peor que otro. Además, el tiempo de ejecución es muy corto. Sin embargo hay escenarios particulares donde la medida ε-calidad no es apropiada.

Como ya hemos comentado anteriormente, los métodos de evaluación, a diferencia de los algoritmos genéticos monobjetivo al aportar más de una solución, trabajan con la dispersión, diversidad y la forma del hipervolumen generado por los individuos de la población. Otras medidas de calidad conocidas son:

- Dados dos conjuntos de soluciones Pareto no dominadas X' y X'', la siguiente función da como resultado un factor dentro del intervalo [0, 1] para determinar cual de los dos conjuntos es mejor que el otro:

$$\boxed{C(X', X'') := |\{a'' \in X''; \exists a' \in X' : a' \prec= a''\}|/|X''|} \tag{34}$$

  $C(X', X'')$ mide el grado de dominancia de X' sobre X''. Véase que $C(X', X'') \neq C(X'', X')$. Es muy similar a la métrica de cobertura, comentada más adelante.

- Distancia al Pareto optimal:

$$\boxed{\begin{aligned} M_1(X') &= \frac{1}{|X'|} \sum_{a' \in X'} min\{\| a' - \overline{a} \|_H; \overline{a} \in \overline{X}\} \\ M_1^*(X') &= \frac{1}{|X'|} \sum_{p' \in X'} min\{\| p' - \overline{p} \|_H; \overline{p} \in \overline{Y}\} \end{aligned}} \tag{35}$$

- Distribución de soluciones no dominadas: Es igual sobre los objetivos

$$\boxed{M_2(X') = \frac{1}{|X' - 1|} \sum_{a' \in X'} |\{b' \in X'; \| a' - b' \| > \sigma\}|} \tag{36}$$

- Extensión de la frontera:

$$\boxed{\begin{aligned} M_3(X') &= \sqrt{\sum_{i=1}^{m} max\{\| a'_i - b'_i \|_H; a', b' \in X'\}} \\ M_3(Y') &= \sqrt{\sum_{i=1}^{n} max\{\| p'_i - q'_i \|; p', q' \in Y'\}} \end{aligned}} \tag{37}$$

- Generación total de vectores no dominados (ONVG): Mide el número de soluciones en PF_{known} y su valor ideal es el máximo posible.
- Razón de la generación total de vectores no dominados (ONVGR): Indica la razón entre la cantidad de vectores no dominados encontrados por cada algoritmo y la cantidad de vectores no dominados existentes en $\mathcal{PF}^*$ y el valor ideal es 1.
- Número de soluciones no dominadas verdaderas (N): Es la cantidad de vectores no dominados propuestos que en efecto pertenecen al frente Pareto óptimo real y su valor ideal es el máximo posible.
- Error máximo del frente Pareto (ME): Indica una banda de error máxima cuando se considera PF_{known} con respecto a $\mathcal{PF}^*$ y su valor ideal es 0.
- Cobertura (C): Representa la relación del número de vectores en el espacio objetivo, encontrados por un algoritmo dado, que son mejores que los encontrados por otro y su valor ideal, en promedio, oscila entre el menor valor de soluciones cubiertas y el mayor valor de cobertura.

Métricas que trabajan con solo un frente:

- HRS, descrita por Collete y Siarry (2002).
- Número de soluciones en el frente óptimo.
- Hipersuperficie, descrita por Collete y Siarry (2002).

Métricas que trabajan con dos frentes:

- Métrica de progresión, por Collete y Siarry (2002).
- Métrica de Laumanns et al. (2000).

Otras métricas que se pueden clasificar son las siguientes:

- Métricas que evalúan la proximidad al frente de Pareto

  - Razón del error (ER): Introducido por Veldhuizen (1999), indica la proporción de vectores en la aproximación del frente de Pareto en el espacio objetivo que no son miembros del frente de Pareto óptimo, y su valor ideal es 0 (la solución i está en el frente de Pareto). Si su valor es 1, la solución i no está en el frente de Pareto. Sin embargo, esta métrica no funciona en el caso de que todas las soluciones de los dos conjuntos comparados no estén en el frente de Pareto. En ese caso, un umbral es utilizado, como si la distancia de la solución i al frente de Pareto fuese mayor que el umbral. En ese caso, $e_i = 1$, en caso contrario vale 0.

$$\boxed{ER = \frac{\sum_{i=1}^{N} e_i}{N}} \tag{38}$$

    donde N es el tamaño del conjunto obtenido.

  - La distancia generacional es una mesura de la cercanía del conjunto de las soluciones no dominadas encontradas en el verdadero frente de Pareto y es calculado como

$$\boxed{GD = \sqrt{\frac{\sum_{i=2}^{n} d_i^2}{n}}} \tag{39}$$

    donde n es el número de puntos del conjunto de soluciones no dominadas y d_i es la distancia euclídea entre cada solución al punto más cercano del verdadero frente de Pareto. En resumen, representa que tan lejos se encuentra PF_{known} de $\mathcal{PF}^*$ y su valor ideal es 0. Si hay una alta fluctuación en los valores de la distancia, es necesario calcular la varianza de la métrica. Los valores objetivo deben ser normalizados antes de calcular la distancia.

- Métricas que evalúan la diversidad entre soluciones no dominadas

  - Espaciado (Schott, 1995): Es una mesura de la distribución del conjunto de soluciones no dominadas encontradas y es calculada como

$$\boxed{SP = \frac{1}{n-1}\sqrt{\sum(\overline{d} - d_i^2)}} \tag{40}$$

    donde n equivale al número de puntos en la solución no dominada; $d_i = min(|f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)|)$, i, $j = 1, ..., n$; y d es la media de todos los d_i. En resumen, sirve como indicador de la distribución de las soluciones en PF_{known} y su valor ideal es 0 (cuanto más pequeño sea, mejor es la distribución en el conjunto). Para algunos problemas, esta métrica debe ser correlacionada con el número de soluciones obtenidas. En general, esta métrica se enfoca en la distribución del conjunto Pareto óptimo, no el grado de dispersión. En Deb et al. (2002), los autores propusieron otro método para mitigar el problema del método de espaciado. La dispersión de un conjunto de individuos no dominadas se calcula del siguiente modo:

$$\boxed{\Delta = \frac{\sum_{i=1}^{M} d_i^e + \sum_{i=1}^{N} |d_i - \overline{d}|}{\sum_{i=1}^{M} d_i^e + N\overline{d}}} \tag{41}$$

    donde d_i puede ser cualquier medida de distancia entre la vecindad de las soluciones y $\overline{d}$ es el valor medio de estas distancias. d_i^e es la distancia entre las soluciones extremas del conjunto no dominado obtenido y el verdadero conjunto Pareto óptimo. El rango de Δ oscila entre 0 y 1. Si es cercano a 1, la dispersión es mala.

- Métricas que evalúan tanto la proximidad como diversidad: Las métricas anteriores de las secciones previas se enfocan en un único criterio. Esta sección resume dos métricas que están relacionadas tanto la diversidad como la cercanía.

- El primero es el ratio de hipervolumen (Zitzler y Thiele, 1998), uno de los más aceptados en la comunidad de los MOEAs. Para calcular el hipervolumen, el área de espacio objetivo cubierta por el frente de Pareto obtenido es calculado, llamada hiperárea. Hay que tener en cuenta que este cálculo requiere tiempo de computación (aunque hay recientes intentos serios de acelerar este proceso (While, Bradstreet, Barone y Hingston, 2005; While, Hingston, Barone y Huband, 2006). En general, considerando dos conjuntos de soluciones, cualquiera que tenga un mejor valor de hipervolumen será el mejor. Sin embargo, cuando usamos el hipervolumen, a veces es difícil entender la calidad del frente Pareto óptimo obtenido en comparación con el verdadero frente Pareto óptimo.

  Como se recomienda en Coello et al. (2002) y Veldhuizen (1999), es considerado mejor el uso del ratio del hipervolumen (HR) que es medido a través del ratio entre los hipervolúmenes de hiperáreas cubiertas por el frente de Pareto óptimo obtenido y el verdadero frente de Pareto óptimo, llamado $H1$ y $H2$ respectivamente. HR es calculado como se muestra en la siguiente ecuación. Para esta métrica, cuanto mayor sea el valor de HR, el algoritmo tiene mejor convergencia.

  $$\boxed{HR = \frac{H_1}{H_2}} \tag{42}$$

  Hay varias cuestiones sobre como determinar el punto de referencia para los cálculos del hipervolumen. Por ejemplo, puede ser el origen (Veldhuizen, 1999). Sin embargo, generalmente es dependiente del área del espacio objetivo que fue visitado comparando todos los algoritmos. En esta versión revisada, como fue sugerida (Deb, 2001), el punto de referencia es un asociado con todos los peores valores objetivos encontrados por todos los algoritmos.

- La segunda métrica usa un método de comparación estadística. Fue introducida por Fonesca y Fleming (1996). Para los experimentos de los MOEAs, que genera un gran conjunto de soluciones, esta métrica suele ser muy eficaz. Knowles y Corne (2000) modificó esta métrica y en vez de dibujar líneas paralelas, todas las líneas son generadas desde el origen. La idea básica es la siguiente: suponemos que los dos algoritmos ($A1, A2$) da como resultado dos conjuntos no dominados: $P1$ y $P2$. Las líneas que se juntan las soluciones en $P1$ y $P2$ son llamadas superficies de logro. La comparación se realiza en el espacio objetivo. El orden para realizar la comparación, el número de líneas son dibujadas desde el origen (asumiendo un problema de minimización) de tal manera que intersectan con las superficies. La comparación entonces es realizada individualmente para cada línea de muestra para determinar cual de ellas supera al resto. Cada línea de intersección produce un número de puntos de intersección. En este caso, los tests estadísticos (como desviación estándar, media, mediana) son necesarios para determinar el porcentaje del algoritmo supera a los otros en cada sección. Para estos dos métodos, los resultados finales son dos números que muestran el porcentaje del espacio donde cada algoritmo supera a otro.

6. Implementación y casos de estudio

Para implementar las distintas pruebas, se han utilizado jMetal (librería usada en los congresos internacionales de estrategias evolutivas para la optimización de problemas), MOEAFramework (framework distribuido como software de calidad para el estudio de estrategías evolutivas multi-objetivo), Platypus (desarrolado por un programador de jMetal) y DEAP. El ordenador donde se han realizado las pruebas es un HP Pavilion 15-p033ns con Arch Linux como sistema operativo. La versión de Java utilizada ha sido la 8 (OpenJDK8) y todo el código de Java ha sido desarrollado en el IDE Eclipse Photon. Para representar las gráficas y mapas han sido utilizadas las librerías mpl_toolkit y matplotlib de Python, mientras que para obtener los datos para los casos de estudio se ha utilizado el paquete googlemaps, la API de Google Maps para Python.

Los datos que se han cogido con referencia en esta primera prueba es la distancia y el tiempo que se tarda de ir de una ciudad a otra, mediante el *script* ~/python/data/generate_data.py. Las ciudades que se han escogido se han codificado de la siguiente manera:

0. Gerona
1. Barcelona
2. Tarragona
3. Tortosa
4. Castellón
5. Valencia
6. Alicante
7. Murcia
8. Almería
9. Málaga
10. Marbella
11. Gibraltar
12. Cádiz
13. Huelva
14. Badajoz
15. Cáceres
16. Salamanca

El *script* genera una matriz de 17x17 donde cada casilla contiene una tupla (distancia, tiempo). Las ciudades se han numerado de 0 a 16, por lo que, por ejemplo, la casilla 5,13 empezando a numerar las casillas por 1 tiene la información de ir de la ciudad 6 a la ciudad 14. Consideraremos que ir de una ciudad i a una ciudad j tiene el mismo coste de objetivos que ir de la ciudad j a la i, por lo que nos encontraremos con una matriz simétrica. Las casillas 1,1; 2,2; ...; 17,17 al minimizar los dos objetivos, se han penalizado esos valores asignándoles tanto para la distancia y el tiempo un valor de 999999999, aunque al usar permutaciones, nunca vamos a tener el problema de tener en la ristra una ciudad repetida y por lo tanto, esos valores nunca se van a utilizar, pero se ponen en caso de que se quiera utilizar otra representación del TSP.

Los casos de estudio con los que se han probado las librerías han sido dos:

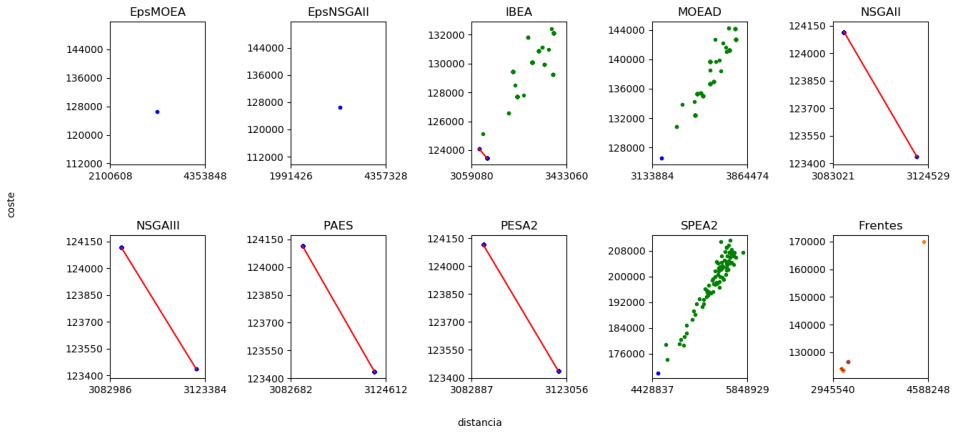
- Considerar la distancia y el tiempo que se tarda en ir entre las ciudades. Como hemos comentado anteriormente en el trabajo, se tratan variables independientes ya que no es lo mismo recorrer 200 kilómetros a través de la meseta de la península ibérica que recorrer 200 kilómetros para atravesar el Himalaya (lógicamente, esa misma distancia se tarda más en recorrer en el Himalaya que en la meseta de la Península Ibérica, ya que en el Himalaya el terreno es accidentado mientras que en la meseta no hay apenas desnivel), en la mayoría de casos la distancia y el tiempo son variables proporcionales, es decir, que para recorrer 200 kilómetros, en la mayoría de casos se tardará aproximadamente un tiempo similar. En estos casos donde las variables son más o menos proporcionales entre sí lo que se suele hacer es solo optimizar un objetivo (algoritmo genético monobjetivo) debido al hecho de que en estos escenarios, normalmente al optimizar un objetivo se suele llegar al óptimo del mismo modo que ejecutando el problema con un algoritmo genético multiobjetivo, con la única diferencia que los algoritmos genéticos multiobjetivo comúnmente tienen una carga computacional más elevada que los algoritmos genéticos monobjetivo. Se han escogido estos datos para comprobar el funcionamiento de los algoritmos genéticos multiobjetivo en estos casos y comprobar que funcionan correctamente, además que las ciudades se han escogido a propósito porque se puede deducir de forma sencilla cuales serán las soluciones óptimas (debería ser una o muy pocas, lo normal es que sea la permutación [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16], visualizado en la siguiente figura). El ficheros donde se encuentran estos datos al generarse son: ~/python/cost.txt; ~/python/distance.txt y ~/python/tsp_python.txt (este último es una variable en Python donde se han guardado los datos de cost.txt y distance.txt). Los dos primeros ficheros son los datos dispuestos entre separaciones y saltos de línea simulando una matriz y se pueden visualizar con un bloc de notas. Además, estos dos últimos ficheros también son necesarios para la implementación en Java.

Figura 34: Mapa generado con el script plot_maps.py
Este recorrido es probablemente el más óptimo para el primer caso de estudio

- Coger la distancia entre las ciudades como primer objetivo y como segundo objetivo se ha multiplicado la distancia por un número aleatorio entre 1 y 0.1 (redondeando el resultado de este último para que sea un entero), como si asignásemos entre el recorrido entre dos ciudades un peaje aleatorio. En este caso no disponemos de las soluciones óptimas *a priori*. Los ficheros de los datos se generan en el mismo directorio que el caso anterior de la misma forma.

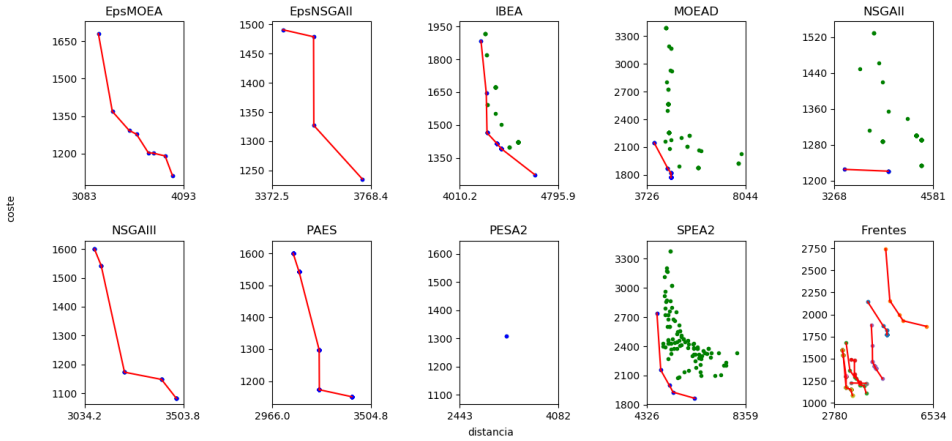
En nuestro ejemplo el espacio de decisión estaría representado por permutaciones, que representarían a cada individuo candidato a pasar a ser una de las soluciones óptimas (no se pueden representar en un gráfico, por lo que las representaremos en mapas, como la figura anterior), mientras que el espacio objetivo, al tratarse de un problema bidimensional, se trataría de una gráfica bidimensional con puntos que representan los valores de cada una de las funciones a optimizar, en este caso la distancia y el coste/valor aleatorio.

Hemos probado el funcionamiento de Platypus en el primer caso (ejecutando el *script* ~/python/platypus_test.py con los ficheros de datos en ese mismo directorio y mediante Python2) con los operadores de cruce y mutación por defecto (aunque se pueden utilizar operadores como el PMX, SBX, ...), con un criterio de minimización en los dos objetivos (esto en todos los casos) y 4000 iteraciones y el resultado es el siguiente:

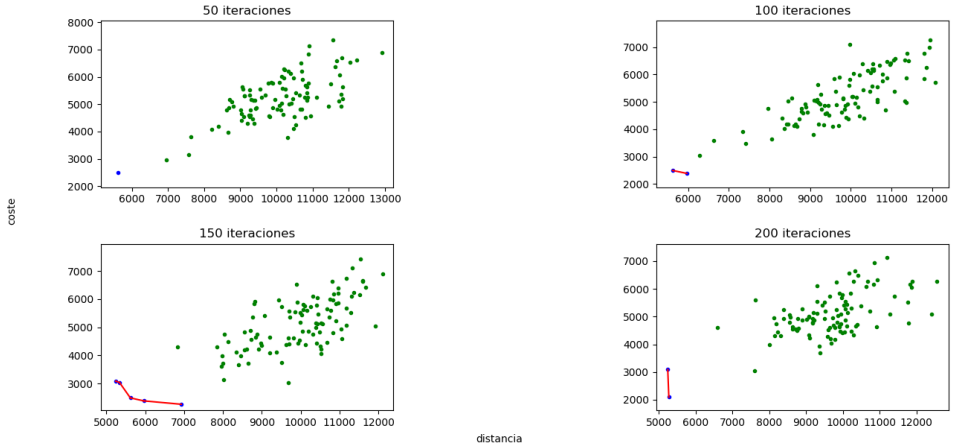


De estos resultados se puede deducir que la librería funciona, ya que si tomamos como ejemplo la gráfica del SPEA2, se observan las soluciones óptimas en color azul y el resto de la población en color verde. Los datos toman una especie de distribución lineal, cosa de suponer ya que, como hemos comentado anteriormente, se ha cogido a optimizar la distancia y el tiempo, y estos más o menos son proporcionales. Para una mayor resolución de los resultados, se recomienda ejecutar el *script* y ampliar la gráfica de forma interactiva. Los resultados del *script* se pueden consultar ya que se crea un directorio ~/python/platypus/ con las permutaciones y su *fitness* (esto lo hacen todas las implementaciones de las librerías), además de la figura anterior.

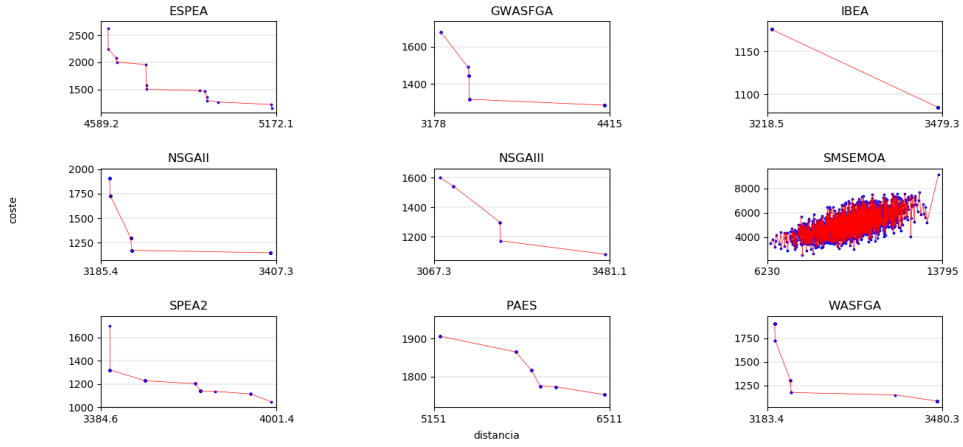
Una vez analizado el primer caso, vamos a aplicar con el segundo caso todas las implementaciones de la librería y los vamos a visualizar. Con Platypus se realiza lo mismo como se ha hecho en el primer caso, y su resultado es el siguiente:



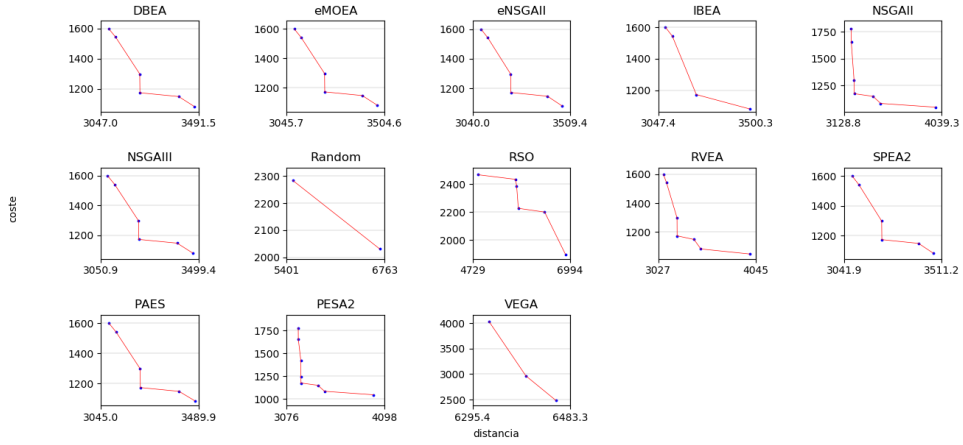
Los resultados de DEAP, utilizando SPEA2 (solo admite ese junto a NSGA-II), muestran la evolución cada 50, 100, 150 y 200 iteraciones. En este caso se ha utilizado un operador de cruce PMX y una probabilidad de mutación de barajado de 0.05. Para comprobarlo, ejecutamos el *script* ~/python/deap_test.py, y al igual que con Platypus, con los ficheros de datos en ese mismo directorio.



Respecto a las implementaciones de jMetal y MOEAFramework, comentar que en el directorio ~/python/data/ hay dos *scripts* para dibujar las soluciones de ambos, además de un tercero (~/python/data/plot_compare_fronts.py) para comparar frentes óptimos de cualquiera de las implementaciones de todas las librerías. Los parámetros para los algoritmos de jMetal son: cruce PMX con probabilidad 0.9, mutación de intercambio con probabilidad 0.5 e 10000 iteraciones. Los resultados de jMetal son los siguientes:



Los de MOEAFramework, con un tamaño de población de 100, aplicando primero un cruce SBX con probabilidad 1 y índice de distribución de 15 y luego otro cruce PMX de probabilidad 0.05, índice de distribución 20 y 30000 iteraciones:



MOEAFramework además imprime las métricas de evaluación de cada algoritmo y los compara con el resto. El resultado que imprime de las ejecuciones de arriba son las siguientes:

- VEGA: Hypervolume: Min: 0.0; Median: 0.0; Max: 0.0; Count: 50; Indifferent: [DBEA, NSGAIII, RSO, Random]
- DBEA: Hypervolume: Min: 0.0; Median: 0.0; Max: 0.0; Count: 50; Indifferent: [VEGA, NSGAIII, RSO, Random]
- NSGAIII: Hypervolume: Min: 0.0; Median: 0.0; Max: 0.31764986864103184; Count: 50; Indifferent: [VEGA, DBEA, RSO, Random]
- eNSGAII: Hypervolume: Min: 0.0; Median: 0.06485159146056321; Max: 0.6794098580440057; Count: 50; Indifferent: [RVEA, SPEA2, NSGAII, IBEA]
- RVEA: Hypervolume: Min: 0.0; Median: 0.08928195113727364; Max:0.744814381448782 ; Count: 50; Indifferent: [eNSGAII, SPEA2, NSGAII, IBEA]
- PESA2: Hypervolume: Min: 0.0; Median: 0.6487647841558404; Max: 0.7596823300612339; Count: 50; Indifferent: [eMOEA, PAES]
- SPEA2: Hypervolume: Min: 0.0; Median: 0.07227352369177908; Max: 0.7429297967097037; Count: 50; Indifferent: [eNSGAII, RVEA, NSGAII, IBEA]
- RSO: Hypervolume: Min: 0.0; Median: 0.0; Max: 0.0; Count: 50; Indifferent: [VEGA, DBEA, NSGAIII, Random]
- eMOEA: Hypervolume: Min: 0.0; Median: 0.5728003644734703; Max: 0.7596823300612339; Count: 50; Indifferent: [PESA2, PAES]
- NSGAII: Hypervolume: Min: 0.0; Median: 0.10549259111335002; Max: 0.690131318827985; Count: 50; Indifferent: [eNSGAII, RVEA, SPEA2, IBEA]
- Random: Hypervolume: Min: 0.0; Median: 0.0; Max: 0.0; Count: 50; Indifferent: [VEGA, DBEA, NSGAIII, RSO]
- IBEA: Hypervolume: Min: 0.0; Median: 0.031001318607213616; Max: 0.7429297967097037; Count: 50; Indifferent: [eNSGAII, RVEA, SPEA2, NSGAII]
- PAES: Hypervolume: Min: 0.0; Median: 0.5582555448737889; Max: 0.7596823300612339; Count: 50; Indifferent: [PESA2, eMOEA]

Por último, comentar que hay un *script* ~/python/data/plot_maps.py para dibujar en un mapa soluciones de los algoritmos. Hay que dejar las soluciones en un archivo llamado ~/python/data/sample y se ejecuta el *script* para visualizar el mapa de cada individuo.

7. Conclusiones

De los resultados obtenidos de las ejecuciones de los algoritmos genéticos multiobjetivo que hemos implementado podemos deducir las siguientes conclusiones, teniendo en cuenta lo siguiente:

- Hay varias métricas de evaluación comunes en los algoritmos genéticos multiobjetivo como ya hemos comentado anteriormente, al igual que hay varias métricas comunes que se pueden utilizar para comparar varias heurísticas, sin embargo, no entraremos más de lleno en estos aspectos de los algoritmos genéticos al no ser uno de los objetivos del trabajo.
- Incluso considerando los parámetros mencionados en el punto anterior, de normal se suele hacer una media de estos parámetros de varias ejecuciones de un algoritmo genético, ya que al tratarse de un método heurístico, varias ejecuciones del algoritmo genético pueden reflejar soluciones muy dispares entre si.

Con esto se obtiene la conclusión de que solo comparando los resultados de una ejecución de cada algoritmo genético no se trata de una medida confiable de que un algoritmo sea mejor que otro. Sin embargo nosotros lo compararemos de esta forma. Se deja pendiente comparar el tiempo de ejecución de cada algoritmo genético multiobjetivo en cada librería para no solo comparar que algoritmos son más rápidos, sino que librería puede ser más óptima para el problema del viajante.

Al ejecutar varias veces los algoritmos en las diversas librerías en las que se han implementado, se ha llegado a la conclusión de que se necesitan entre 3000-4000 iteraciones para que todos los algoritmos lleguen a converger hacia soluciones que el mayor número de veces se tratan de soluciones óptimas. Durante las varias ejecuciones se ha observado:

- El algoritmo PAES en la librería Platypus es el que más rápido converge (necesita menos iteraciones) hacia la solución óptima, por lo que en principio se trataría del algoritmo más adecuado para este tipo de problema en esa librería. Excepto el SPEA2 y MOEA/D, el resto de soluciones óptimas de los algoritmos son muy parecidas. Puede que el tema de la búsqueda local tenga algo que ver con ello.
- En jMetal, el NSGA-III parece ser el que aporta mejor resultados.

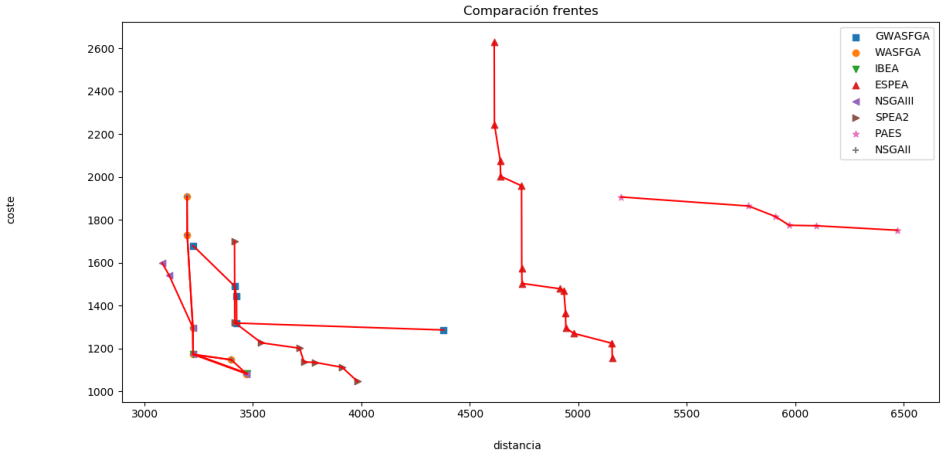


Figura 35: Comparación frentes resultados jMetal

- En MOEAFramework, excepto los algoritmos Random, RSO y VEGA, el resto de frentes a simple vista parecen ser similares.

Hay que comentar que en todas las implementaciones no se puede especificar la ciudad de inicio ni la de fin, por lo que en caso de que se quisiese especificar, se tendría que modificar la función de *fitness*/evaluación/aptitud de los individuos penalizando aquellos individuos que no tuviesen la ciudad de inicio ni de fin en el sitio correcto.

Como posibles extensiones o ampliaciones del trabaja desarrollado pueden considerarse:

- La mayoría de la literatura de la resolución del TSP en algoritmos evolutivos están centrados en las búsquedas locales. Haría falta realizar un estudio de estas técnicas y si se pueden combinar con los algoritmos comentados para ver si se puede realizar alguna mejora.
- Analizar otras posibles mejoras locales en los algoritmos genéticos multiobjetivo para ver si puede dar mejores resultados en el TSP.
- La representación de problemas multiobjetivo con restricciones es otro de los temas el cual estaría bien estudiar ya que en la mayoría de problemas los algoritmos genéticos están generalizados para soportar restricciones bastante simples en el problema, por lo que un estudio de problemas con restricciones más complejas y estrictas sería un buen tema para profundizar más en la temática de los algoritmos genéticos multiobjetivo.
- Adaptar la librería para que resuelva otros problemas multiobjetivo básicos como el problema de la mochila.

Las principales dificultades que se han tenido a lo largo del trabajo son las siguientes:

- Se intentó desarrollar un algoritmo genético monobjetivo desde cero para luego modificar el paso de selección y los que hiciesen falta para desarrollar por separado los algoritmos genéticos multiobjetivo. Sin embargo, aunque se consiguió desarrollar un algoritmo genético monobjetivo desde cero (~/python/mono/), las distintas formas de operar de cada algoritmo genético multiobjetivo y el tiempo necesario para implementarlas hizo que se tomase un cambio de rumbo y utilizar librerías que ya los tenían implementados. A la larga es mejor, además de que se pueden combinar con librerías de algoritmos genéticos monobjetivo para no implementar métodos de cruce o mutación.
- Además, indicar que hay pocos datos de problemas TSP multiobjetivo. Se ha encontrado una página web donde en principio hay datos sobre un TSP biobjetivo con la solución óptima, sin embargo no se han podido interpretar los datos de los ficheros [BIT18]. Para problemas multiobjetivo donde hay que optimizar varias funciones matemáticas si que hay una gran variedad de *benchmarks* disponibles que suelen utilizarse a la hora de analizar la eficiencia de un algoritmo genético multiobjetivo, entre los más utilizados se encuentran los problemas ZDT (Zitzler, E., Deb, K., and Thiele, L.)/DTLZ/DTLZ2 (Deb-Thiele-Laumanns-Zitzler).

Bibliografia

[ABI18] Genetic Algorithm for Traveling Salesman Problem with Modified Cycle Crossover Operator, Abid Hussain, Yousaf Shad Muhammad, M. Nauman Sajid, Ijaz Hussain, Alaa Mohamd Shoukry and Showkat Gani, emitido el 11 de mayo del 2018. Consultado a `https://www.hindawi.com/journals/cin/2017/7430125/`.

[AHM15] A multi-objective evolutionary algorithm using decomposition (MOEA/D) and its application in multipurpose multi-reservoir operations, I. Ahmadianfar, A. Adib, M. Taghian. International journal of optimization in civil engineering, Int. J. Optim. Civil Eng., emitido el 12 de marzo del 2015; 5(2): 167-187. Consultado a `https://www.researchgate.net/publication/277017307_A_multi-objective_evolutionary_algorithm_using_decomposition_moead_and_its_application_in_multipurpose_multi-reservoir_operations`.

[AWA15] Multiobjective Optimization, Mariette Awad, Rahul Khanna. Efficient Learning Machines pp 185-208. Consultado a `https://link.springer.com/chapter/10.1007/978-1-4302-5990-9_10`.

[BIO13] Bioinformática 2013-2014. Tema 12-MOEA, apuntes de la Universidad Politécnica de Valencia, emitido en 2013. Consultado a `http://docplayer.es/58090682-Bioinformatica.html`.

[BIT18] Biobjective TSP, emitido el 6 de mayo del 2018. Consultado a `https://eden.dei.uc.pt/~paquete/tsp/`.

[BRA08] Jürgen Branke. Multi-objective Optimization Inspired by Nature, consultado el 25 de agosto de 2018. Consultado a `http://www.lamsade.dauphine.fr/~projet_cost/ALGORITHMIC_DECISION_THEORY/pdf/Branke.pdf`.

[CIR16] Guillermo Campos Ciro, Frédêric Dugardin, Farouk Yalaoui, Russell Kelly. A NSGA-II and NSGA-III comparision for solving an open shop scheduling problem with resource constraints.

[COE01] Carlos A. Coello Coello, Tutorial on Evolutionary Multiobjective Optimization, emitido en marzo del 2001. Consultado a `https://www.cs.cinvestav.mx/~emooworkgroup/tutorial-slides-coello.pdf`.

[COE02] Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 1st edition, 2002.

[COE07] Carlos A. Coello Coello, Gary B. Lamont, David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edition, 2007.

[COR00] David W. Corne, Joshua D. Knowles, Martin J. Oates. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization.

[COR01] David W. Corne, Nick R. Jerram, Joshua D. Knowles, Martin J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization.

[COR08] Carlos Adrián Correa Flórez, Ricardo Andrés Bolaños, Alexánder Molina Cabrera. Algoritmo multiobjetivo NSGA-II aplicado al problema de la mochila.

[DEA18] DEAP, emitido el 10 de septiembre del 2018. Consultado a `https://deap.readthedocs.io/`.

[DEB01] Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Springer, 1st edition, 2001.

[DEM18] Wikipedia, Deme (biology), emitido el 9 de abril del 2018. Consultado a `https://en.wikipedia.org/wiki/Deme_(biology)`.

[DUA01] Susana Duarte Flores. Optimización multiobjetivo de redes empleando algorítmos evolutivos paralelos.

[GHA16] Rahman Gharari, Navid Poursalehi, Mohammadreza Abbasi, and Mahdi Aghaie. Implementation of Strength Pareto Evolutionary Algorithm II in the Multiobjective Burnable Poison Placement Optimization of KWU Pressurized Water Reactor.

[ISG18] Universidad del Pais Vasco, Intelligent Systems Group, emitido el 28 de julio de 2018. Consultado a http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf.

[IVO01] Ivo F. Sbalzarini, Petros Koumoutsakos. Multiobjective Optimization Using Evolutionary Algorithms.

[JME18] jMetal 5 Web Page, emitido el 10 de septiembre del 2018. Consultado a https://jmetal.github.io/jMetal/.

[KNO99] Joshua Knowles, David Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation.

[LAM08] Lam Thu Bui, Sameer Alam. An Introduction to Multi-Objective Optimization.

[LAR99] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza y S. Dizdarevic. Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators. Consultado a https://www.researchgate.net/publication/226665831_Genetic_Algorithms_for_the_Travelling_Salesman_Problem_A_Review_of_Representations_and_Operators?_sg=1y1fv85z82qtjLIXVpsUIZ8sRI4GiHcIBl3C40SykXNxs9KgJ6KiftItLBrfB3GfgUa4vNJfJA.

[LEC01] Lecture 9: Multi-Objective Optimization, consultado el 24 de agosto de 2018. Consultado a https://engineering.purdue.edu/~sudhoff/ee630/Lecture09.pdf.

[LUC04] Christian von Lücken, Augusto Hermosilla, Benjamín Barán. Algoritmos Evolutivos para Optimización Multiobjetivo: un Estudio Comparativo en un Ambiente Paralelo Asíncrono.

[MEN07] César Augusto Peñuela Meneses, Mauricio Granada Echeverri. Optimización multiobjetivo usando un algoritmo genético y un operador elitista basado en un ordenamiento no-dominado (NSGA-II).

[MOF18] MOEA Framework, emitido el 10 de septiembre del 2018. Consultado a http://moeaframework.org/.

[MOP18] Wikipedia, Multi-objective optimization, emitido el 12 de abril del 2018. Consultado a https://en.wikipedia.org/wiki/Multi-objective_optimization.

[PLA18] Platypus, emitido el 10 de septiembre del 2018. Consultado a https://platypus.readthedocs.io/.

[RUD15] Filip Rudziński. Finding Sets of Non-Dominated Solutions with High Spread and Well-Balanced Distribution using Generalized Strength Pareto Evolutionary Algorithm.

[SES06] Aravind Seshadri. A fast elitist multiobjective genetic algorithm: NSGA-II.

[TSP18] Wikipedia, Problema del viajante, emitido el 16 de julio del 2018. Consultado a https://es.wikipedia.org/wiki/Problema_del_viajante.

[YUA09] Yuan Ze, Mitsuo Gen. Network Models and Optimization: Multiobjective GA Approach, consultado el 25 de agosto de 2018. Consultado a http://logistics.iem.yzu.edu.tw/Gen/seminarII-YZU-IEM-MGen.pdf.

[YVA11] Dr. Yván Jesús Túpac Valdivia. Algoritmos Evolutivos en Optimización Multiobjetivos (MOEA).

[ZHA07] MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, Qingfu Zhang and Hui Li. IEEE Transactions On Evolutionary Computation, Vol. 11, No. 6, emitido el 6 de diciembre del 2007. Consultado a https://dces.essex.ac.uk/staff/zhang/papers/moead.pdf.

[ZIT01] Eckart Zitzler, Marco Laumanns, Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm.

[ZIT02] Eckart Zitzler. Evolutionary Algorithms For Multiobjective Optimization.

[ZIT04] Eckart Zitzler, Marco Laumanns, and Stefan Beuler. A Tutorial on Evolutionary Multiobjective Optimization.