Revista Iberoamericana de Automática e Informática Industrial
CEA IFAC comité español de automática
RIAI
ISSN: 1697-7912. Vol. 4, Núm. 1, Enero 2007, pp. 5-23
http://riai.isa.upv.es

UNA REVISIÓN DEL ESTADO DEL ARTE EN OPTIMIZACIÓN.

José A. Caballero[1] Ignacio E. Grossmann[2]

[1]*Departamento de Ingeniería Química Universidad de Alicante (España)*
[2]*Departamento de Ingeniería Química. Carnegie Mellon University. (EEUU)*

Resumen: En este artículo se hace una revisión de las técnicas de optimización más importantes: programación lineal y no lineal con y sin variables discretas, así como de los algoritmos disponibles hasta la fecha para resolver dichos problemas. Además de los problemas clásicos se hace una extensión para incluir problemas planteados mediante disyunciones y relaciones lógicas, optimización global determinista y estocástica, optimización en problemas sin estructura definida y una visión global de los sistemas de modelado algebraico.

Palabras Clave: Optimización, LP, NLP, MINLP, MILP, Programación disyuntiva, Optimización global.

1. INTRODUCCIÓN

Hoy en día, la optimización se ha convertido en práctica habitual en las ciencias, las ingenierías y los negocios. En estadística son comunes términos como "*máxima probabilidad o mínimos cuadrados*" en el campo de los negocios es frecuente encontrar términos como "*máximo beneficio, mínimo coste o aprovechamiento óptimo de recursos*". En física se han enunciado diferentes principios óptimos en campos como la óptica o la mecánica clásica, etc. Aunque, muchos de los aspectos básicos de la optimización se desarrollaron en los siglos XVIII y XIX con los trabajos de Lagrange o Euler el verdadero desarrollo de la *Programación Matemática* comienza con los trabajos de Kantorovich y Dantzing en los años cuarenta. Sin embargo, no es hasta la revolución informática de los años 70 cuando, apoyados en el poder de cálculo de los ordenadores, la programación matemática comienza a ser una herramienta ampliamente utilizada en muchas aplicaciones. De hecho hoy en día existen una cantidad importante de códigos que cada vez pueden resolver problemas mas grandes y complejos. Para una panorámica general de herramientas de optimización se recomienda que el lector consulte la pagina NEOS una vez haya leído este artículo: http://www-fp.mcs.anl.gov/otc/Guide/

Se intentará en este artículo dar una visión general de los métodos más importantes de optimización. Las primeras secciones están dedicadas a una revisión general de los principales métodos para resolver problemas con variables continuas. A continuación se revisan con algo más de detalle los métodos más importantes para resolver problemas cuando aparecen variables discretas. Una de las grandes limitaciones de la programación matemática cuando aparecen ecuaciones no lineales es que solamente se puede garantizar la optimalidad global cuando se cumplen ciertas condiciones (función objetivo convexa definida sobre una región factible –restricciones- también convexa). Se introducirá por lo tanto un breve apartado dedicado a las técnicas de optimización global. Para tratar con problemas sin una estructura especial (funciones discontinuas, ecuaciones donde no hay información de las derivadas, restricciones definidas de forma implícita a través de relaciones de tipo caja negra entrada-salida o problemas altamente no convexos) se han desarrollado métodos de búsqueda directa o metaheurísticos que, aunque no garantizan un óptimo global, tienden a aproximarse a éste si el número de iteraciones es suficientemente alto. Finalmente se presentan los principales sistemas de modelado disponibles hoy en día y que se han convertido en una herramienta imprescindible para formular modelos complejos.

Los problemas de optimización se pueden, en primer lugar, clasificar en problemas que involucran solamente variables continuas y en problemas con variables continuas y discretas (o solamente discretas). Los principales problemas de optimización con variables continuas incluyen la Programación

Lineal (LP) y la Programación no lineal (NLP)[i] . Una importante subclase de la programación lineal es el problema lineal complementario (LCP) y subproblemas importantes de programación no lineal incluyen la programación cuadrática (QP). En el caso de programación no lineal hay dos distinciones importantes a tener en cuenta. La primera es si el problema no lineal es convexo y la segunda es si el problema es diferenciable (Biegler y Grossmann, 2004). Si el problema incluye variables discretas se puede nuevamente distinguir entre problemas lineales con variables enteras (MILP) y problemas no lineales con variables enteras (MINLP) ambas siglas del ingles *Mixed Integer (non)Linear Programming*. Cabe señalar que para el caso en el que el problema contenga ecuaciones diferenciales/algebraicas estas se pueden discretizar por varios métodos (por ejemplo colocación ortogonal) lo que da lugar a su vez a problemas NLP o MINLP.

El caso más general de problema de programación matemática podría pues escribirse como:

$$\begin{aligned} &\min: z = f(x,y) \\ &s.a.\ h(x,y) = 0 \\ &\quad g(x,y) \le 0 \\ &\quad x \in X \subseteq \Re^n \\ &\quad y \in \{0,1\}^m \end{aligned} \qquad (1)$$

Donde *f* representa una función objetivo escalar, *h* y *g* son restricciones del problema en forma de igualdades y desigualdades, las variables *x* son variables continuas y las variables *y* son variables binarias (en general las variables *y* podrían tomar cualquier conjunto de valores discretos, pero restringirlas a solamente variables binarias no quita generalidad al problema y será el criterio que seguiremos de aquí en adelante).

2. PROGRAMACIÓN LINEAL

Si tanto la función objetivo como las restricciones son lineales entonces el problema tiene la estructura siguiente:

$$\begin{aligned} &\min: z = c^T x \\ &s.a.\ A x \le b \\ &\quad x \ge 0 \end{aligned} \qquad (2)$$

El método estándar para resolver un problema lineal es el método simplex, desarrollado por Dantzing en los años cuarenta. Vèase (Dantzing, 1963). Las mejoras posteriores en la implementación práctica del método simplex junto con el desarrollo de los ordenadores ha hecho que hoy en día se puedan resolver de forma cotidiana problemas de más de 10^5 restricciones y un millón de variables. Se ha observado que el número de iteraciones requeridas para resolver un LP utilizando el método simplex es habitualmente un múltiplo pequeño del número de restricciones del problema. Sin embargo, es posible plantear problemas donde sea necesario examinar todas las bases. Así en un problema con *n* variables y *m* restricciones, el número de iteraciones en el peor caso sería de $\binom{n}{m}$, por ejemplo un problema con 100 variables y 50 restricciones –un problema que podemos considerar hoy en día pequeño- en un hipotético ordenador capaz de llevar a cabo un billón (10^{12}) de iteraciones del simplex por segundo tardaría unos $4 \cdot 10^{14}$ años en resolver el problema. Aunque este comportamiento se observa en rarísimas ocasiones, existía un interés tanto teórico como práctico en desarrollar un algoritmo para resolver los problemas de programación lineal en tiempo polinomial. En 1984 se presentó un algoritmo de punto interior con esta propiedad (Karmarkar, 1984). Con los métodos de punto interior se pueden resolver problemas de más de 10^5 variables y 10^5 restricciones utilizando entre 20 y 40 iteraciones independientemente del tamaño del problema. Dado que los métodos de programación lineal son ampliamente conocidos, en particular el método simplex, no se hará más énfasis en ellos

3. PROGRAMACIÓN NO LINEAL

Un problema de programación no lineal (NLP) toma la siguiente forma:

$$\begin{aligned} &\min: f(x) \\ &s.a.\ h(x) = 0 \\ &\quad g(x) \le 0 \\ &\quad x \in X \subseteq \Re^n \end{aligned} \qquad (3)$$

Dentro de los algoritmos para resolver NLPs aquellos que requieren un menor número de evaluaciones de la función objetivo son los basados en la programación cuadrática sucesiva (SQP –*successive quadratic programming*- (Han, 1976; Powell, 1978; Binder et al., 2001; Biegler y Grossmann, 2004) que ha dado lugar a una serie de algoritmos que se adaptan muy bien a un gran número de problemas con diferentes estructuras.

La SQP se basa en aplicar el método de Newton a las condiciones de optimalidad de Karush –Khun-Tucker (KKT) del problema (3), que vienen dadas por las siguientes ecuaciones:

[1] Aunque en castellano para programación lineal y programación no lineal, en ocasiones se utilizan las siglas PL y PNL, para los casos donde aparecen variables binarias los problemas MILP, MINLP las siglas en ingles se han convertido en estándar por lo que se ha decido, para mantener una cierta coherencia, utilizar las correspondientes siglas en inglés durante todo el artículo.

$$\begin{aligned}
&\nabla f(x^*) + \nabla h(x^*)\lambda + \nabla g(x^*)\mu = 0 \\
&h(x^*) = 0 \\
&g(x^*) + s = 0 \qquad (4)\\
&SMe = 0 \qquad (c1)\\
&(s,\mu) \geq 0 \qquad (c2)
\end{aligned}$$

Donde e = [1,1,1,…, 1]^T, λ es el vector de multiplicadores de Lagrange asociado a las restricciones de igualdad y μ el vector de multiplicadores de Lagrange de las desigualdades. S = diag{s}; M = diag{ μ }.

Las condiciones de complementaridad –ecuaciones (c1) y (c2) del problema (4)- presentan una importante dificultad cuando se resuelve el conjunto de ecuaciones asociado a las condiciones de KKT: Cerca de la solución las ecuaciones (c1) y los límites activos (c2) son dependientes y por lo tanto forman un sistema de ecuaciones mal condicionado. Los algoritmos basados en SQP solucionan este problema de dos formas diferentes: utilizando un método de conjunto activo o utilizando funciones de barrera.

En los métodos de conjunto activo en cada iteración se debe decidir qué subconjunto de restricciones son activas, $i \in I = \{i \mid g_i(x^*) = 0\}$. En la ecuación (4.c1) se obliga a que $s_i = 0 \ \forall i \in I$ y $\lambda_i = 0 \ \forall i \notin I$. La determinación del conjunto activo es un problema de carácter combinatorio. Sin embargo, una forma relativamente sencilla de determinar el conjunto activo en un punto x^k, consiste en resolver el siguiente problema cuadrático. (Un problema cuadrático es un problema con función objetivo cuadrática y restricciones lineales, y se puede resolver de forma muy eficiente utilizando técnicas de programación lineal, ya que el problema cuadrático se puede convertir en un problema lineal complementario):

$$\begin{aligned}
&\min \ \nabla f(x^k)^T d + \frac{1}{2} d^T L(x^k, \lambda^k, \mu^k) d \\
&s.a. \ h(x^k) + \nabla h(x^k)^T d = 0 \qquad (5)\\
&\quad g(x^k) + \nabla g(x^k)^T d + s = 0 \quad s \geq 0
\end{aligned}$$

Las condiciones de optimalidad de KKT para el problema (5) vienen dadas por:

$$\begin{aligned}
&\nabla f(x^k) + W(x^k, \lambda^k, \mu^k) d + \nabla h(x^k)\lambda + \nabla g(x^k)\mu = 0 \\
&h(x^k) + \nabla h(x^k)^T d = 0 \\
&g(x^k) + \nabla g(x^k)^T d + s = 0 \\
&SMe = 0 \\
&(s,\mu) \geq 0
\end{aligned} \qquad (6)$$

Donde $W(x,\lambda,\mu) = \nabla^2 \left(f(x) + h(x)^T \lambda + g(x)^T \mu \right)$ es la matriz Hessiana de la función de Lagrange. Es fácil comprobar que las tres primeras ecuaciones en (6) corresponden a linealizaciones de las ecuaciones en (5). La solución del problema combinatorio de elegir el conjunto activo viene dada por la solución de problema cuadrático (6). Detalles de este procedimiento se pueden encontrar en (Fletcher, 1987; Nocedal y Wright, 1999)

Para evitar el problema combinatorio de seleccionar el conjunto activo, los métodos de barrera modifican el NLP de la siguiente manera:

$$\begin{aligned}
&\min \ f(x^k) - \rho \sum_i Ln(s_i) \\
&s.a. \ h(x^k) = 0 \qquad (7)\\
&\quad g(x^k) + s = 0 \\
&\quad s \geq 0
\end{aligned}$$

Donde la solución de este problema toma un $s > 0$ para el parámetro $\rho > 0$. Las condiciones de KKT del problema (7) se pueden escribir como:

$$\begin{aligned}
&\nabla f(x^*) + \nabla h(x^*)\lambda + \nabla g(x^*)\mu = 0 \\
&h(x^*) = 0 \\
&g(x^*) + s = 0 \qquad (8)\\
&SMe = \rho e
\end{aligned}$$

Para valores de $\rho > 0, s > 0 \ y \ \mu > 0$ las iteraciones del método de Newton generadas al resolver el problema (8) están bien condicionadas. Si además W^k es definida positiva en la proyección sobre el espacio nulo de $\nabla h(x^k)^T$, el paso de Newton se puede escribir en forma del siguiente subproblema cuadrático:

$$\begin{aligned}
&\min : \nabla f(x^k)^T d + \frac{1}{2} d^T W(x^k, \lambda^k, \mu^k) d - \\
&\quad - \rho (S^k)^{-1} e^T \Delta s + \frac{1}{2} \Delta s^T (S^k)^{-1} \Delta s \qquad (9)\\
&s.a. \ h(x^k) + \nabla h(x^k)^T d = 0 \\
&\quad g(x^k) + \nabla g(x^k)^T d + s^k + \Delta s = 0
\end{aligned}$$

Las condiciones de complementaridad en el problema (9) han sido reemplazadas por penalizaciones en la función objetivo y por lo tanto el carácter combinatorial del problema desaparece.

Los métodos de barrera necesitan más iteraciones para resolver el problema (7) para diferentes valores del parámetro de penalización (ρ). Los métodos de conjunto activo por su parte, llevan a cabo la solución de subproblemas cuadráticos mas costosos. Así pues, en aquellos problemas donde hay un número pequeño de desigualdades los métodos de conjunto activo suelen dar mejores resultados. Sin embargo, en problemas con muchas desigualdades los métodos de barrera suelen funcionar mejor. Esto es especialmente cierto en los problemas de gran escala donde el número de límites activos suele ser grande. Por ejemplo, una implementación eficiente de los

métodos de barrera en el código IPOPT ha permitido la solución de problemas con más de 2 10^6 variables y 4500 grados de libertad (Wächter, 2002).

Para mejorar la convergencia desde puntos relativamente alejados de la solución se han utilizado, tanto en los métodos de conjunto activo como en los de punto interior, dos estrategias diferentes: búsqueda unidireccional o el método de la región de confiabilidad. En éste último caso a los problemas cuadráticos se les añade la restricción $\|d\| \le \Delta$. De tal forma que el paso $x^{k+1} = x^k + d$ se lleva a cabo sólo si se produce una reducción suficiente de una función de mérito (suma ponderada de la función objetivo y alguna medida de la violación de las restricciones del problema). Dos funciones de mérito habituales en SQP son la Lagrangiana aumentada (LA) y la penalización exacta (PE)

$$\begin{aligned} \Psi(x,\lambda,\mu) &= f(x) + \lambda^T h(x) + \mu^T g(x) \\ &\quad + \rho\|g(x)_+ + h(x)\|^2 \qquad (LA) \\ \Psi(x,\lambda,\mu) &= f(x) + \lambda^T h(x) + \mu^T g(x) \\ &\quad + \rho\|g(x)_+ + h(x)\| \qquad (PE) \end{aligned} \tag{10}$$

El tamaño de la región de confiabilidad Δ se actualiza dependiendo del valor de la función de mérito en cada iteración (Nocedal y Wright, 1999). El código KNITRO (Byrd et al, 1997) utiliza un método de región de confiabilidad. Estos métodos son especialmente adecuados para NLPs mal condicionados.

Por su parte la búsqueda unidireccional es más eficiente en problemas con subproblemas cuadráticos bien condicionados y puntos iniciales razonables. En estos casos el paso resultante de resolver el problema cuadrático se modifica de tal manera que: $x^{k+1}{}' = x^k + \alpha d$ con $\alpha \in [0,1]$. El valor de α se elige de tal forma que se produzca un descenso suficiente en la función de mérito. Recientemente ha aparecido una alternativa basada en una minimización de la función objetivo y de la factibilidad de las restricciones como objetivos en competencia. Este tipo de búsqueda unidireccional se conoce como 'filtro' y también es aplicable a los métodos de región de confiabilidad (Fletcher et al, 2002; Wätcher y Biegler, 2002)

Finalmente remarcar que para que los subproblemas cuadráticos tengan solución única es necesario que la matriz Hessiana de la Lagrangiana sea de rango completo y definida positiva cuando se proyecta en el espacio nulo de los gradientes de las restricciones activas. Estas propiedades podrían no cumplirse en puntos lejos de la solución en problemas que no satisfacen las condiciones de optimalidad de segundo orden. En estos casos se pueden utilizar aproximaciones de la Hessiana reducida (como la BFGS). Véase, por ejemplo, (Fletcher, 1987).

Además de los métodos basados en programación cuadrática sucesiva se han desarrollado otros métodos muy robustos para problemas a gran escala. Aunque en general estos métodos necesitan un mayor número de evaluaciones de la función objetivo y de las restricciones que los métodos basados en SQP, su robustez en un buen número de problemas y el hecho de que han sido implementados en plataformas de modelado muy populares como GAMS o AMPL han contribuido a su difusión. Todos estos métodos se basan también en la aplicación del método de Newton a una parte de las condiciones de KKT (Biegler y Grossmann, 2004).

LANCELOT (Conn et al., 2000) utiliza una Lagrangiana aumentada, de forma que en cada iteración resuelve el siguiente problema:

$$\begin{aligned} \min:\ & f(x) + \lambda^T h(x) + \mu^T (g(x)+s) + \\ & + \frac{1}{2}\rho\,\|h(x) + g(x) + s\|^2 \qquad (11) \\ s.a.\quad & s \ge 0 \end{aligned}$$

El problema (11) se puede resolver de forma muy eficiente para valores fijos de de los multiplicadores λ, μ y del parámetro de penalización ρ. Una vez que se resuelve el problema (11) los multiplicadores y el parámetro de penalización se actualizan en un bucle externo. El procedimiento se repite hasta que se satisfacen las condiciones de KKT.

En MINOS (Murtagh y Saunders, 1987), en contraste con el SQP, el subproblema cuadrático se reemplaza por un NLP con restricciones lineales y una Lagrangiana aumentada en la función objetivo no lineal:

$$\begin{aligned} \min\ & f(x) + \lambda^T h(x) + \mu^T (g(x)+s) + \frac{1}{2}\rho\|h(x), g(x)+s\|^2 \\ s.t.\ & h(x^k) + \nabla h(x^k)^T\, d = 0 \\ & g(x^k) + \nabla g(x^k)^T\, d + s = 0 \quad s \ge 0 \end{aligned} \tag{12}$$

En cada iteración MINOS selecciona una base y trabaja en el espacio reducido de variables de decisión. Después de eliminar las variables dependientes y las variables en alguno de sus límites se aplica un método quasi-Newton al problema sin restricciones resultante. La descomposición está hecha de tal manera que en el caso de que el problema sea lineal MINOS se convierte en el método simplex. En el punto solución de este subproblema se vuelve a linealizar y el ciclo se repite hasta que se cumplen las condiciones de optimalidad. MINOS es extremadamente eficiente en problemas donde la mayor parte de las restricciones son lineales.

Finalmente los métodos de gradiente reducido generalizado (GRG2, CONOPT, SOLVER) consideran el mismo subproblema que MINOS, pero eliminando los términos debidos a la Lagrangiana

aumentada (la función objetivo se reduce a f(x)). En cada iteración, introducen una etapa de restauración de la factibilidad asegurando así que el problema es factible en todo momento mientras se acerca a la solución. Aunque en general utilizan más evaluaciones de función objetivo y restricciones que MINOS suelen ser mucho más robustos en problemas con restricciones no lineales. De entre todos los métodos que utilizan derivadas los métodos de gradiente reducido son los más populares. En particular el código SOLVER se ha incluido dentro del paquete MS Excel alcanzando un uso ampliamente generalizado.

4. PROGRAMACIÓN LINEAL MIXTA (MILP)

Un MILP se puede escribir de forma general como:

$$\begin{aligned} &\min : c^T x + d^T y \\ &s.a \quad Ax + By \le b \\ &\quad x \in X \subseteq \Re^n \\ &\quad y \in \{0,1\}^m \end{aligned} \tag{13}$$

Los métodos para resolver MILPs (Nemhauser y Wolsey, 1988) están fundamentalmente basados en los métodos de ramificación y acotamiento (BB del Inglés *Branch and Bound*) y sus variantes, donde cada subproblema lineal se resuelve utilizando el método simplex (Dankin, 1965). Este método consiste en una enumeración en árbol en el cual el espacio de variables enteras se divide de forma sucesiva dando lugar a subproblemas lineales que se resuelven en cada nodo del árbol.

En el nodo inicial las variables enteras se relajan como variables continuas, de tal forma que se les permite tomar valores fraccionarios. Si la solución de este problema produce de forma natural una solución en la cual todas las variables *y* toman valores enteros se habría alcanzado la solución. Sin embargo, esto sólo ocurre en un número muy reducido de casos (por ejemplo en los problemas de asignación) y lo normal es que algunas de las variables *y* tomen valores fraccionarios. En cualquier caso, este nodo inicial produce una cota inferior global al óptimo del problema. De entre las variables que han tomado valores fraccionarios se debe seleccionar una de ellas para ramificar de acuerdo a una serie de reglas predeterminadas (coste reducido, parte decimal más cercana a 0.5, etc). Una vez que se ha elegido una variable para fijar se resuelve el nuevo problema LP con la variable selecciona fija. Cuando se aplica el método simplex la solución de este LP se puede actualizar de forma muy eficiente a partir del resultado de su nodo antecesor. Esta propiedad no la comparten los algoritmos de punto interior, por lo que en los métodos para resolver MILPs los algoritmos de punto interior prácticamente no se utilizan. El procedimiento continua ramificando nodos abiertos (nodos en los que se ha obtenido una solución factible con alguna variable *y* no entera). Para la elección del nodo a ramificar se utilizan reglas heurísticas (búsqueda en profundidad, búsqueda en anchura, etc…). Es importante remarcar que un nodo cualquiera es una cota inferior para todos los nodos posteriores generados a partir de éste, es decir, a medida que descendemos por las ramas del árbol la función objetivo de los nodos va creciendo de forma monótona. Cuando en un nodo se obtiene una solución entera, ésta es un límite superior a la solución óptima del problema, de tal forma que todas las ramas abiertas con valor superior de la función objetivo no necesitan evaluarse (acotamiento). La enumeración continúa hasta que la diferencia entre las cotas inferior y superior están dentro de una tolerancia o bien no existen ramas abiertas.

En el peor de los casos, el algoritmo básico de ramificación y acotamiento termina con la enumeración de todos los nodos del árbol. Dos desarrollos importantes han contribuido a mitigar el crecimiento exponencial en la solución de MILPs: El desarrollo de técnicas de pre-procesamiento y la introducción de planos de corte. El pre-procesamiento se basa en la utilización de técnicas de eliminación automática de variables y restricciones, reducción de límites, reformulación de restricciones y fijar a priori algunas variables enteras. Los planos de corte son restricciones extra añadidas al problema, bien en el nodo inicial o dentro de la enumeración, que tienen el efecto de reducir la región factible del problema sin comprometer ninguna de las soluciones enteras del mismo. Los últimos desarrollos en MILP incluyen los métodos de ramificación y precio (Barnhart et al, 1998) y ramificación y corte (Balas et al, 1993). Una revisión reciente de los métodos de optimización se puede encontrar en (Johnson et al., 2000).

Los principales códigos para resolver MILPs están desarrollados sobre programas originalmente dedicados a problemas lineales. Quizás los que mejores resultados ofrecen hoy en día son CPLEX (ILOG, 2000), XPRESS (Ashford y Daniel, 1995) y OSL (Ming et al, 1993). Aunque es posible que debido a la naturaleza combinatoria de los problemas MILP, en algún caso el tiempo de cálculo para resolver algún problema sea grande, es importante señalar que estos códigos han experimentado mejoras impresionantes en sus capacidades en los últimos años debido a una combinación en el uso de planos de corte (por ejemplo cortes de Gomory), mejoras de preprocesado y aumento en la velocidad de cálculo de los ordenadores (Bixby et al, 2002)

5. PROGRAMACIÓN NO LINEAL CON VARIABLES ENTERAS Y CONTINUAS (MINLP)

Los principales métodos para resolver MINLPs son:
1. Ramificación y acotamiento (Borchers y Michell, 1994; Gupta y Ravindran, 1985; Leyfer 2001; Stubbs y Mehrotra, 1999) que no son más que una extensión directa de los métodos de ramificación y acotamiento

empleados para resolver MILPs, con la diferencia de que en cada nodo se debe resolver un NLP y la resolución del los diferentes NLPs partiendo de su inmediato antecesor no es tan eficiente como cuando se resuelve un problema lineal. 2. Métodos de descomposición que iteran entre dos subproblemas. El primero es un NLP con valores fijos de las variables binarias y por lo tanto es una cota superior a la solución optima del problema y un problema Maestro – que suele ser un MILP- y que es una cota inferior a la solución óptima del problema. Los métodos más importantes son la descomposición de Benders Generalizada (Geofrion, 1972) y el método de las aproximaciones exteriores (Duran y Grossmann, 1986; Fletcher y Leyfer, 1994; Yuan et al, 1988). 3. LP-NLP basado en ramificación y acotamiento (Quesada y Grossmann 1992), es una situación intermedia entre los métodos de ramificación y acotamiento y los métodos de descomposición. En este caso en lugar de resolver el problema Maestro hasta optimalidad, tan pronto como se encuentra una solución entera se resuelve un NLP y se actualizan todos los nodos abiertos del árbol de búsqueda. 4. Plano de corte extendido (Westerlund y Pettersson, 1995) es una variación que no requiere la solución de NLPs

Los métodos anteriores tienen en común que están basados en resolver una serie de subproblemas obtenidos a partir de la formulación general del problema. Se presentarán a continuación cada uno de estos sub-problemas y verá luego como la combinación de éstos da lugar a los distintos métodos para resolver MINLPs.
Un MINLP se puede formular de forma general como

$$\begin{aligned} &\min : z = f(x,y) \\ &s.a. \quad g_j(x,y) \le 0 \quad j \in J \\ &\quad x \in X \subseteq \Re^n \\ &\quad y \in \{0,1\}^m \end{aligned} \qquad \text{(P1)}$$

Donde se asume que f(·) g(·) son convexas y diferenciables, X esta dado por el conjunto convexo $X = \{x \mid x^L \le x \le x^U,\ Ax \ge b\}$ y J es un índice de desigualdades.

5.1 Subproblemas NLP

Hay tres sub-problemas básicos que se pueden considerar para el problema (P1):

a) <u>NLP-Relajado (NLP1)</u>

$$\begin{aligned} &\min : Z_{LB}^k = f(x,y) \\ &s.a. \quad g_j(x,y) \le 0 \qquad j \in J \\ &\quad x \in X, \quad y \in Y_R \\ &\quad y_i \le \alpha_i^k \quad i \in I_{FL}^k \\ &\quad y_i \ge \beta_i^k \quad i \in I_{FU}^k \end{aligned} \qquad \text{(NLP1)}$$

donde Y_R es la relajación continua del conjunto Y; I_{FL}^k, I_{FU}^k son subconjuntos de índices de las variables enteras, que están restringidas por límites superiores e inferiores α_i^k, β_i^k, en el paso k-ésimo de un procedimiento de enumeración en ramificación y acotamiento. Se debe señalar que $\alpha_i^k = \lfloor y_i^l \rfloor$, $\beta_i^k = \lceil y_i^m \rceil$, $l < k$, $m < k$ donde y_i^l, y_i^m son los valores no enteros de una etapa previa, y que $\lfloor \cdot \rfloor, \lceil \cdot \rceil$ son los operadores de redondeo por defecto y por exceso, respectivamente.
Señalar también que si $I_{FL}^k = I_{FU}^k = \varnothing$, (*k=0*), el problema (NLP1) corresponde a la relajación continua del problema (P1). Excepto para un pequeño número de casos especiales, la solución de este problema produce, en general, un vector no entero de variables discretas. El problema (NLP1) también corresponde al k-ésimo paso en una búsqueda por ramificación y acotamiento. El valor óptimo del objetivo del problema NLP1, Z_{LB}^0 proporciona un límite inferior absoluto al valor óptimo del objetivo del problema (P1) para cualquier $m \ge k$, el límite es sólo válido si $I_{FL}^k \subset I_{FL}^m$, $I_{FU}^k \subset I_{FU}^m$.

b) <u>Sub-problema NLP para un valor fijo de y^k. (NLP2)</u>

$$\begin{aligned} &\min : Z_U^k = f(x,\ y^k) \\ &s.a. \quad g_j(x, y^k) \le 0 \qquad j \in J \\ &\quad x \in X \end{aligned} \qquad \text{(NLP2)}$$

Cuando el problema NLP2 tiene una solución factible el valor de Z_U^k es una cota superior a la solución óptima del problema (P1). O dicho de otra manera, si el problema NLP2 tiene solución factible, corresponde a una posible solución del problema P1, es por lo tanto una cota superior al mismo. En el caso de que el problema NLP2 fuera no-factible deberíamos considera el siguiente sub-problema:

c) <u>Problema de factibilidad para un valor fijo de y^k. (NLPF)</u>

$$\begin{aligned} &\min : u \\ &s.a. \quad g_j(x,\ y^k) \le u \quad j \in J \\ &\quad x \in X, \quad u \in \Re^1 \end{aligned} \qquad \text{(NLPF)}$$

El problema NLPF se puede interpretar como la minimización de la norma infinito correspondiente a la no-factibilidad del sub-problema (NLP2). Para un (NLP2) no-factible la solución del problema (NLPF) da un valor de *u* estrictamente positivo.

5.2. Planos de corte MILP

La convexidad de las funciones no lineales se puede explotar reemplazándolas con hiperplanos soporte que se obtienen generalmente, aunque no

necesariamente, de la solución de los sub-problemas NLP. En particular, los nuevos valores y^k o del par (x^k, y^k) se obtienen resolviendo un problema de plano de corte (MILP) que está basado en los K puntos (x^k, y^k), k= 1…K generados en los K pasos previos:

$$\begin{aligned} &\min \ Z_L^k = \alpha \\ &s.a. \ \ \alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k)\begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ &g_j(x^k, y^k) + \nabla g_j(x^k, y^k)\begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0 \quad j \in J^k \\ &x \in X \ , \ \ y \in Y \qquad\qquad k = 1.... K \end{aligned} \qquad \text{(M-MIP)}$$

donde $J^k \subseteq J$. Cuando sólo se incluye un subconjunto de linealizaciones, suelen corresponder a las restricciones violadas del problema (P1). Alternativamente, es posible incluir todas las linealizaciones en el problema (M-MIP). La solución del (M-MIP) es un límite inferior válido Z_L^k del problema (P1). Este límite inferior es no decreciente con el número de puntos K linealizados. Señalar que dado que las funciones f(·) y g(·) son convexas, las linealizaciones en (M-MIP) corresponden a aproximaciones exteriores de la región no lineal factible del problema (P1). Una interpretación geométrica se muestra en la Figura 1, donde se puede observar que la función objetivo convexa es subestimada, mientras que la región factible convexa es sobreestimada por las linealizaciones.

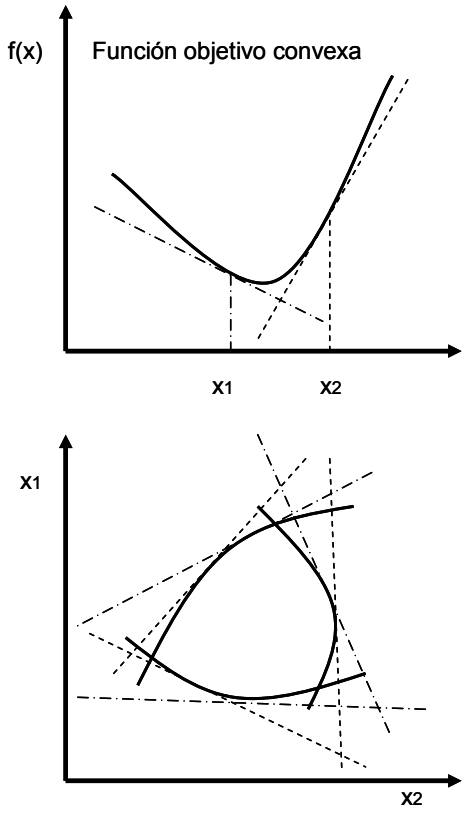

Figura 1. Interpretación geométrica de las linealizaciones en el problema (M-MIP)

5.3 Algoritmos

Los diferentes métodos se pueden clasificar de acuerdo al uso que se hace de los sub-problemas (NLP1), (NLP2) y (NLPF), y de la especialización del problema MILP (M_MIP), como se indica en la Figura 2.

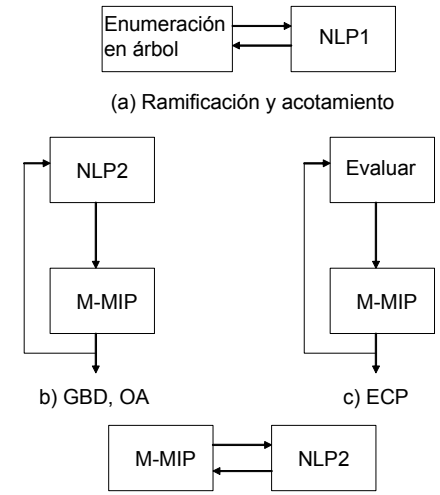

Figura 2. Principales pasos en los diferentes algoritmos MINLP

Remarcar que en la descomposición de Benders generalizada (GBD) en el método de las aproximaciones exteriores (OA) –caso b- y en el método LP/NLP basado en ramificación y acotamiento –caso d- se debe resolver un NLPF en lugar del correspondiente NLP si este sub-problema fuere no-factible. Cada uno de los métodos se explica a continuación en términos de los sub-problemas básicos que los forman.

Ramificación y Acotamiento

Aunque los primeros trabajos en ramificación y acotamiento estaban orientados a problemas lineales, este método se puede aplicar también a problemas no lineales. El método de ramificación y acotamiento (BB) comienza resolviendo el (NLP1) con $I_{FL}^k = I_{FU}^k = \varnothing$, ($k=0$) -problema de relajación continua- si todas las variables discretas toman valores enteros entonces la búsqueda se detiene, se habrá localizado el óptimo del problema. En otro caso, se comienza una búsqueda en árbol en el espacio de las variables enteras $y_i \ i \in I$. Estas variables se van fijando sucesivamente en los correspondientes nodos del árbol dando lugar a problemas NLP relajados de la forma (NLP1) los cuales producen límites inferiores para los sub-problemas de los nodos descendientes. La eliminación de un nodo y sus descendientes ocurre cuando el límite inferior sobrepasa al actual límite superior, cuando el sub-problema es no-factible o cuando todas las variables enteras toman valores

discretos. Esta última situación produce un límite superior a la solución del problema.

Los métodos de ramificación y acotamiento son sólo atractivos si los sub-problemas NLP son relativamente fáciles de resolver o cuando sólo es necesario resolver una pequeña parte de ellos. Esto podría ser así o bien porque la dimensionalidad, referida al número de variables discretas es pequeña, o bien porque la relajación continua produce una cota inferior muy buena. (La diferencia con la solución óptima no es grande).

Método de las Aproximaciones Exteriores (OA)

El método de las aproximaciones exteriores (OA, del inglés *Outer Approximation*) surge cuando se resuelven de forma sucesiva subproblemas (NLP2) y problemas Maestro MILP (M-MIP) en un ciclo de iteraciones para generar puntos (x^k, y^k).

El algoritmo de las aproximaciones exteriores tal como fue propuesto originalmente por Duran y Grossmann consiste en efectuar una seria de iteraciones k=1,…K. Se comienza resolviendo el problema NLP1 de relajación continua (o bien con un conjunto pre-especificado de variables discretas y^0). La solución óptima de este problema se utiliza para generar el primer problema Maestro (RM-OA). La solución del problema Maestro genera un nuevo conjunto de variables y^k (en la primera iteración k=1). Este nuevo conjunto de variables discretas se utiliza para resolver un nuevo NLP2 o bien, si este no es factible, un NLPF. Las linealizaciones de la solución de este problema se añaden al problema Maestro, continuando así de forma iterativa.

Los subproblemas (NLP2) producen un límite superior que se utiliza para definir la mejor solución obtenida en un momento dado, $UB^k = \min_k \{Z_U^k\}$. El ciclo de iteraciones continúa hasta que los límites superior e inferior del problema Maestro están dentro de una tolerancia especificada.
Una forma para evitar resolver los problemas de factibilidad (NLPF) en el algoritmo de las aproximaciones exteriores cuando el problema (P1) está planteado en función de variables binarias 0-1, consiste en introducir un corte entero cuyo objetivo es hacer no-factible la elección de las combinaciones 0-1 generadas en iteraciones anteriores:

$$\sum_{i \in B^k} y_i - \sum_{i \in N^k} y_i \leq |B| - 1 \qquad k = 1....K \qquad \text{(ICUT)}$$

donde $B^k = \{i \mid y_i^k = 1\}$, $N^k = \{i \mid y_i^k = 0\}$. Este corte se convierte en muy débil a medida que aumenta el número de variables binarias. Sin embargo, tiene la propiedad de asegurar que en sucesivas iteraciones se van a generar nuevas combinaciones de variables binarias (sin repetir ninguna de las anteriores). Utilizando el corte entero anterior la finalización del algoritmo tiene lugar tan pronto como $Z_L^K \geq UB^K$.
El método de las aproximaciones exteriores generalmente necesita un número pequeño de iteraciones para converger.

Es también importante remarcar que el problema Maestro no necesita ser resuelto hasta optimalidad. De hecho, dado un límite superior UB^k y una tolerancia ε, es suficiente generar el nuevo punto (x^k,y^k).

Descomposición de Benders Generalizada (GDB)

La descomposición de Benders Generalizada (GDB, del inglés Generalizad Benders Decomposition) es similar al método de las aproximaciones exteriores. Véase (Flippo y Kan, 1993). Las diferencias surgen en la definición del problema Maestro MILP (M-MILP). En GDB sólo se consideran las desigualdades activas: $J^k = \{j \mid g_j(x^k, y^k) = 0\}$ y el conjunto $x \in X$ se ignora. En particular, considere las linealizaciones del método de las aproximaciones exteriores en un punto $(x^k,y^{k)}$),

$$\begin{aligned} &\alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ &g_j(x^k, y^k) + \nabla g_j(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq 0 \end{aligned} \qquad (\text{OA}^k)$$

Donde para un punto fijo y^k el punto x^k corresponde a la solución óptima del problema NLP2. Haciendo uso de las condiciones de Karush-Kuhn-Tucker y eliminando las variables continuas x, las desigualdades en (OA^k) se pueden reducir a:

$$\begin{aligned} \alpha \geq\, & f(x^k, y^k) + \nabla_y f(x^k, y^k)^T (y - y^k) + \\ & + (\mu^k)^T \left[g(x^k, y^k) + \nabla_y g(x^k, y^k)(y - y^k) \right] \end{aligned} \qquad (\text{LC}^k)$$

que es el corte Lagrangiano proyectado en el espacio *y*. Esto se puede interpretar como una restricción de sustitución de las ecuaciones en (OA^k), porque se obtiene como combinación lineal de éstas.
Para el caso en el que no hay solución factible al problema (NLP2), entonces el punto x^k se obtiene solucionando el problema de factibilidad (NLPF). El siguiente corte de factibilidad, proyectado en el espacio *y* se puede obtener por un procedimiento similar:

$$(\lambda^k)^T \left[g(x^k, y^k) + \nabla_y g(x^k, y^k)(y - y^k) \right] \leq 0 \quad (\text{FC}^k)$$

De esta manera el problema Maestro (M-MIP) se reduce a un problema proyectado en el espacio *y*.

$$\min Z_L^k = \alpha$$
$$s..a. \quad \alpha \geq f(x^k, y^k) + \nabla_y f(x^k, y^k)^T (y - y^k)$$
$$+ (\mu^k)^T \left[g(x^k, y^k) + \nabla_y g(x^k, y^k)(y - y^k) \right] \quad k \in KFS$$
$$(\lambda^k)^T \left[g(x^k, y^k) + \nabla_y g(x^k, y^k)(y - y^k) \right] \leq 0 \quad k \in KIS$$
$$x \in X, \quad \alpha \in \Re^1$$

(RM-GDB)

donde KFS es el subconjunto de los problemas (NLP2) factibles y KIS es el subconjunto de los problemas no factibles cuya solución viene dada por (NLPF). Por supuesto, se debe cumplir que $|KFS \cup KIS| = K$. Dado que el problema Maestro (RM-GBD) se puede obtener a partir del problema Maestro (RM-OA), en el contexto del problema (P1). La descomposición de Benders Generalizada se puede considerar un caso particular del algoritmo de las aproximaciones exteriores. De hecho el límite inferior predicho por el problema relajado (RM-OA) es mayor o igual que el predicho por el problema master relajado (RM-GDB), (Duran y Grossmann, 1986)

Debido al hecho de que los límites inferiores obtenidos por GDB son más débiles que en OA, este método suele requerir un mayor número de iteraciones. A medida que el número de variables 0-1 aumenta este efecto se hace más pronunciado, lo cual se corresponde con lo que cabría esperar dado que en cada iteración sólo se añade un nuevo corte. Habitualmente, el usuario debe añadir restricciones con objeto de reducir los límites. Por otra parte, aunque el método de las aproximaciones exteriores, predice mejores límites inferiores que GBD el coste computacional para resolver el problema maestro (M-OA) es mayor dado que el número de restricciones añadidas por iteración es igual al número de restricciones no lineales más la linealización de la función objetivo.

Método del Plano del Corte Extendido

El método del plano de corte extendido (ECP del inglés *Extended Cutting Plane*) es una extensión del método de plano de corte (Kelley, 1960). No resuelve problemas NLP. Confía simplemente en la solución iterativa de problemas (M-MIP) añadiendo linealizaciones sucesivas a aquella restricción más violada en el punto predicho (x^k, y^k). La convergencia se obtiene cuando la violación máxima cae dentro de una tolerancia especificada. Es, por supuesto, posible añadir linealizaciones a todas las restricciones violadas, o incluso linealizaciones a todas las restricciones no lineales. En el algoritmo ECP la función objetivo debe ser lineal, lo cual se puede conseguir fácilmente introduciendo una nueva variable para transferir las no linealidades de la función objetivo a una desigualdad.

Señalar que, dado que las variables continuas y discretas se convergen simultáneamente, el algoritmo ECP podría requerir un gran número de iteraciones.

LP/NLP basado en Ramificación y Acotamiento (LP/NLP-BB)

Desarrollado por (Quesada y Grossmann, 1992), el método es similar en espíritu a los métodos de ramificación y corte, y evita la solución completa del MILP Maestro (M-OA) en cada iteración. El método comienza resolviendo un subproblema NLP inicial, el cual se linealiza como en (M-OA). La idea básica consiste en efectuar una búsqueda por ramificación y acotamiento, resolviendo LPs, en cada rama del árbol. En cuanto un nodo presenta una solución entera, se resuelve un (NLP2) y se actualiza la representación del problema maestro en todos los nodos abiertos del árbol con la adición de las correspondientes linealizaciones. Se evita así la necesidad de re-comenzar la búsqueda en árbol.

Este método se puede aplicar tanto a los algoritmos GBD como ECP. La búsqueda LP/NLP generalmente reduce bastante significativamente el número de nodos a enumerar. Como contrapartida podría incrementar el número de problemas NLP2 que deberían ser resueltos. Aunque la experiencia con este método muestra que esto no es así y el número total de NLP2 suele permanecer constante.

Este algoritmo puede considerarse en un punto intermedio entre los métodos de descomposición y los métodos de ramificación y acotamiento. De hecho (Bonami, et al., 2005) han propuesto un algoritmo unificado que permite variar desde un BB hasta OA pasando por LP/NLP-BB donde el nivel de actualizaciones del problema Maestro es fácilmente modificable a través de un parámetro.

5.4 Extensiones de los métodos para MINLP

En esta sección presentamos algunos de los extensiones más importantes de los métodos presentados hasta el momento.

Problema master cuadrático.

Para muchos problemas de interés, el problema (P1) es lineal en y: $f(x,y) = \phi(x) + c^T y$; $g(x,y) = h(x) + B y$. Cuando este no es el caso (Fletcher y Leyffer, 1994) sugieren incluir una aproximación cuadrática a (RM-OAF) de la forma:

$$\min \quad Z^k = \alpha + \frac{1}{2}\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix}^T \nabla^2 L\,(x^k, y^k)\begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix}$$

$$s.a.\ \ \alpha \le UB^k - \varepsilon$$

$$\left.\begin{array}{l} \alpha \ge f(x^k, y^k) + \nabla f(x^k, y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} \\ g(x^k, y^k) + \nabla g(x^k, y^k)^T \begin{pmatrix} x-x^k \\ y-y^k \end{pmatrix} \le 0 \end{array}\right\} k = 1....K$$

$$x \in X,\ y \in Y,\ \alpha \in \Re^1 \qquad \text{(M-MIQP)}$$

Señalar que en este caso el problema maestro no produce un límite inferior válido, sin embargo dado que f(·) y g(·) son convexas el método continúa llevando a soluciones rigurosas porque el acotamiento de las aproximaciones exteriores a través de α sigue siendo válido. Cuando la función objetivo es no lineal en *y*, el problema OA original necesita mayor número de iteraciones que cuando se usa el problema maestro cuadrático. El precio que hay que pagar, sin embargo, es que se debe resolver un MIQP en lugar de un MILP.

Reducción de la dimensionalidad del problema Maestro.

El problema maestro (RM-OA) puede llegar a incluir un número bastante grande de restricciones como consecuencia de la acumulación de linealizaciones. Una opción es mantener sólo el último punto linealizado, pero esto puede llevar a la no-convergencia incluso en el caso de problemas convexos, dado que el incremento monotónico de la secuencia de límites inferiores no se puede garantizar. Una forma rigurosa de reducir el número de restricciones sin sacrificar demasiado la calidad del límite inferior se puede conseguir en casos de problemas MINLP mayoritariamente lineales:

$$\begin{array}{l} \min\ Z = a^T w + r(v) + c^T y \\ s.a.\quad D w + t(v) + C y \le 0 \\ \qquad F w + G v + E y \le b \\ \qquad w \in W\ ,\ v \in V\ ,\ y \in Y \end{array} \qquad \text{(PL)}$$

donde (*w, v*) son variables continuas y *r(v) t(v)* son funciones convexas. Como muestran (Quesada y Grossmann, 1992) las aproximaciones lineales de la función objetivo y de las restricciones no lineales se pueden agregar en el siguiente problema maestro:

$$\begin{array}{l} \min\ Z_L^k = a^T w + \beta + c^T y \\ s.a.\ \ \beta \ge r(v^k) + \left(\lambda^k\right)^T \left[D w + t(v^k) + C y\right] + \\ \qquad + \left(\mu^k\right)^T \left[G(v - v^k)\right]\ k = 1....K \\ \qquad F w + G v + E y \le b \\ \qquad w \in W,\ \ v \in V,\ \ y \in Y,\ \ \beta \in \Re^1 \end{array} \qquad \text{(M-MIPL)}$$

Los resultados numéricos muestran que la calidad de los límites no se ve muy afectada por la agregación del anterior MILP en comparación con lo que sucedería si se aplicase GBD.

Tratamiento de restricciones de igualdad.

Para el caso de restricciones lineales que aparezcan como igualdad no hay ninguna dificultad puesto que estas restricciones son invariantes respecto a las linealizaciones. Sin embargo, si las ecuaciones son no lineales aparecen dos dificultades. Primero, no es posible forzar la linealización de igualdades en K puntos diferentes. Segundo, las ecuaciones no lineales introducen, en general, no convexidades, excepto si se pudieran relajar como desigualdades. (Kocis y Grossmann, 1987) propusieron una estrategia de relajación en que las igualdades son reemplazadas por desigualdades:

$$T^k \nabla h(x^k, y^k) \begin{bmatrix} x-x^k \\ y-y^k \end{bmatrix} \le 0$$

donde $T^k = \{t_{ii}^k\}$, $t_{ii}^k = signo(\lambda_i^k)$ y en donde λ_i^k es el multiplicador de Lagrange asociado a la ecuación $h_i(x, y) = 0$. Señalar que si estas ecuaciones se relajan como desigualdades $h(x, y) \le 0$ para todo *y*, y además *h(x,y)* es una función convexa, el procedimiento es riguroso. En otro caso se podrían generar hiperplanos soporte no válidos. Por otra parte, para el problema GBD no hay que tener ninguna precaución especial dado que estas ecuaciones simplemente se incluyen en los cortes Lagrangianos. Sin embargo, si dichas ecuaciones no se relajan como desigualdades convexas tampoco se garantiza la convergencia global de este método.

Tratamiento de no-convexidades.

Cuando *f(x,y), g(x,y)* son funciones no convexas en (P1), o cuando las iguadades *h(x,y)=0,* están presentes, aparecen dos dificultades. Primero, Los subproblemas NLP (NLP1, NLP2, NLPF) podrían no tener un único mínimo local. Segundo, el problema maestro y sus variantes (M-MIP, M-GBD, M-MIQP) no garantizan un límite inferior válido con el consiguiente peligro de cortar el óptimo global del problema. Una posible solución consiste en reformular el problema, sin embargo esta solución está restringida a un número pequeño de problemas.
Por otra parte, es posible utilizar técnicas de optimización global determinista basándose en estructuras especiales de los términos continuos (por ejemplo, bilineales, fraccional lineal, cóncava separable, etc). La idea es desarrollar envolventes convexas o sub-estimadores para formular problemas que sean límites inferiores rigurosos. Más adelante se comentarán algunas de las técnicas de optimización global determinista que se pueden aplicar.

Quizás la opción más utilizada para tratar no convexidades consiste en aplicar una estrategia heurística que trata de reducir tanto como sea posible el efecto de las no convexidades. Aunque es un método no riguroso, requiere mucho menos esfuerzo computacional. A modo de comentario señalar que los métodos de optimización global determinista están restringidos, hoy por hoy, a problemas de pequeño tamaño o con estructuras muy especiales. Describiremos a continuación un método que reduce el efecto de las no convexidades a nivel del problema Maestro. Continúa sin embargo confiando en que el resultado de los NLPs sea el optimo global de cada uno de ellos. (Viswanathan y Grossmann, 1990) propusieron introducir variables de holgura en el problema Maestro (MILP) con objeto de reducir la probabilidad de cortar la región factible. Este problema Maestro se conoce como APER del inglés *Aumented Penalty /Equality Relaxation*). Y toma la forma:

$$\min \; Z_L^k = \alpha + \sum_{k=1}^{K} \left[w_p^k \, p^k + w_q^k \, q^k \right]$$

$$\left.\begin{array}{l} s.a. \; \alpha \geq f(x^k, y^k) + \nabla f(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \\ T^k \, \nabla h(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq p^k \\ g_j(x^k, y^k) + \nabla g_j(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq q^k \\ \sum_{i \in B^k} y_i - \sum_{i \in N^k} y_i \leq |B| - 1 \end{array}\right\} k = 1..K$$

$$x \in X \;, \quad y \in Y \;, \quad \alpha \in \Re^1 \;, \quad p^k, q^k \geq 0 \qquad \text{(M-APER)}$$

donde w_p^k, w_q^k son penalizaciones suficientemente grandes (p.e. 1000 veces la magnitud del multiplicador de Lagrange). Remarcar que si las funciones son convexas entonces el MILP Maestro (M-APER) predice un límite inferior riguroso al problema (P1) dado que todas las variables de holgura toman el valor cero.

Por último, otra modificación posible para reducir el efecto indeseable de las no convexidades en el problema Maestro consiste en aplicar un test de convexidad global seguido por una validación de las linealizaciones. Una posibilidad es aplicar el test a todas las linealizaciones con respecto al actual vector solución (x^k, y^k) (Kravanja y Grossmann, 1994). Las condiciones de convexidad que deben ser verificadas para las linealizaciones son las siguientes:

$$\left.\begin{array}{l} f(x^k, y^k) + \nabla f(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} - \alpha \leq \varepsilon \\ T^k \, \nabla h(x^k, y^k) \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq \varepsilon \\ g_j(x^k, y^k) + \nabla g_j(x^k, y^k)^T \begin{bmatrix} x - x^k \\ y - y^k \end{bmatrix} \leq \varepsilon \end{array}\right\} k = 1..K-1$$

donde ε es un vector de pequeñas tolerancias (p.e. 10^{-10}). El test se omite para el último punto K porque todas las linealizaciones son válidas para este punto. Aquellas restricciones para las cuales no se satisfaga son simplemente eliminadas del problema Maestro. Este test confía en la suposición de que las soluciones de los sucesivos NLPs se van aproximando al óptimo global, y en que las sucesivas validaciones van progresivamente definiendo restricciones válidas alrededor del óptimo global.

5.5 Códigos de ordenador para MINLP

El número de códigos de ordenador para resolver problemas del tipo MINLP es todavía bastante reducido. El programa DICOPT (Viswanathan y Grossmann 1990) está disponible en el sistema de modelado GAMS. El código está basado en el problema master (M-APER) y en los subproblemas (NLP2). Este código utiliza también la relajación continua (NLP1) para generar la primera linealización con lo que el usuario no necesita especificar un valor entero inicial. Dado que las límites inferiores rigurosos no se pueden garantizar con (M-APER) la búsqueda para problemas no convexos termina cuando no se produce mejora en los NLP factibles entre dos iteraciones consecutivas. Esta heurística funciona razonablemente bien en muchos problemas. AIMMS (Bisschop y Roelofs, 1999) ha implementado también una variación del método de aproximaciones exteriores. Algunos códigos que implementan el método de ramificación y acotamiento usando subproblemas (NLP1) incluyen a MINLP_BB que está basado en un algoritmo SQP (Leyffer, 2001) y está disponible en AMPL y El código SBB que también está disponible en GAMS. El código α-ECP implementa los métodos de plano de corte extendido (Westerlund y Pettersson, 1995), incluyendo la extensión de (Pörn y Westerlund, 2000). Finalmente el código MINOPT (Schweiger et al., 1996) también implementa los métodos OA y GBD, y los aplica a optimización dinámica con variable entera. Es difícil hacer comentarios generales sobre la eficiencia y confiabilidad de estos códigos dado que no se ha hecho una comparación sistemática entre ellos. Sin embargo, se podría anticipar que los métodos de ramificación y acotamiento probablemente funcionarían mejor si la relajación de los MNILP fuese buena. Los métodos de descomposición basados en OA probablemente funcionarían mejor si los subproblemas NLP fueran relativamente costosos de resolver, mientras que GBD puede ser eficiente si la relajación del MINLP es buena y aparecen muchas variables binarias. Los

métodos basados en ECP tienden a funcionar mejor en problemas altamente lineales. Por ultimo cabe señalar el código Bonmin que es "*open source*" e implementa el método de rama y acotamiento, aproximaciones exteriores y el método híbrido de (Quesada y Grossmann, 1992) (véase https://projects.coin-or.org/Bonmin)

6. PROGRAMACIÓN DISYUNTIVA GENERALIZADA (GDP)

Recientemente ha aparecido una nueva tendencia que representa los problemas de optimización discretos/continuos con modelos que mezclan restricciones algebraicas, disyunciones lógicas y relaciones lógicas (Balas, 1985; Beaumont, 1991; Raman y Grossmann, 1993, 1994; Turkay y Grossmann, 1996; Hooker y Osorio, 1999; Hooker 2000; Lee y Grossmann 2000). En particular, el MINLP del problema (P1) se puede formular como un problema disyuntivo generalizado de la siguiente manera (Raman y Grossmann, 1994) y que puede ser considerado como una extensión de la programación disyuntiva introducida por (Balas ,1985):

$$
\begin{aligned}
&\min \sum_k c_k + f(x) \\
&s.a.\ g(x) \le 0 \\
&\bigvee_{i \in D_k} \begin{bmatrix} Y_{ik} \\ h_{ik}(x) \le 0 \\ c_k = \gamma_{ik} \end{bmatrix} \quad k \in SD \qquad \text{(GDP)} \\
&\Omega(Y) = Verdad \\
&x \in \Re^n,\ c \in \Re^m,\ Y \in \{Verdad, Falso\}^m
\end{aligned}
$$

En el problema GDP Y_{ik} son variables booleanas que establecen si un término en la disyunción es verdadero, y por lo tanto deben cumplirse las ecuaciones dentro de dicho término. $\Omega(Y) = Verdad$ son relaciones lógicas, en forma de lógica de proposiciones, que involucran solamente a variables booleanas y que expresan relaciones entre los conjuntos de disyunciones.

Es importante remarcar que el problema GDP se puede siempre transformar en un MINLP de la forma del problema P1, y viceversa, cualquier MINLP de la forma P1 se puede transformar en su equivalente disyuntivo. Sin embargo, a efectos de modelado es siempre conveniente y ventajoso comenzar con la representación disyuntiva del modelo dado que captura de forma mucho más directa los aspectos cualitativos y cuantitativos del mismo (Vecchietti y Grossmann, 1999, 2000). La forma más directa de hacer la transformación desde GDP a MINLP consiste en reemplazar las variables boolenas (Y) por variables binarias (*y*) y las disyunciones por restricciones de tipo M grande, de la forma:

$$
\begin{aligned}
&h_{ik}(x) \le M(1 - y_{ik}) \quad i \in D,\ k \in SD \\
&\sum_{i \in D_k} y_{ik} = 1 \quad k \in SD \qquad \text{(M-GDP)}
\end{aligned}
$$

donde M es un límite superior válido a la restricción correspondiente. Las proposiciones lógicas $\Omega(Y) = Verdad$ se pueden convertir en desigualdades algebraicas lineales que dependen sólo de las variables binarias (Williams, 1985; Raman y Grossmann 1991). La mayor limitación de la reformulación de M grande es que la relajación inicial suele ser débil.

Alternativamente es posible reformular el problema utilizando la envolvente convexa (*convex hull*) de las disyunciones (Stubbs y Mehrota, 1999; Grossmann y Lee, 2002)

$$
\begin{aligned}
&\min \sum\sum \gamma_{ik}\, \lambda_{ik} + f(x) \\
&s.a.\ g(x) \le 0 \\
&\quad x = \sum_{i \in D_k} \nu_{ik} \ ,\ \sum_{i \in D_k} \lambda_{ik} = 1 \quad k \in SD \\
&\quad \lambda_{ik}\, h\left(\frac{\nu_{ik}}{\lambda_{ik}}\right) \le 0 \quad i \in D_k \qquad k \in SD \qquad \text{(RDP)} \\
&\quad A\lambda \le a \\
&x \in \Re^n\ ,\ \nu_{i,k} \ge 0,\ 0 < \lambda_{i,k} \le 1\ \ i \in D_k\ \ k \in SD
\end{aligned}
$$

El problema anterior se puede utilizar como base para desarrollar un método específico de ramificación y acotamiento (Ceria y Soares, 1999; Lee y Grossmann, 2000). La idea del algoritmo desarrollado por Lee y Grossmann consiste en ramificar directamente sobre las restricciones correspondientes a términos particulares de cada disyunción mientras se considera el *convex hull* de las restantes disyunciones. Alternativamente es posible resolver el problema anterior como un MINLP simplemente cambiando la última restricción del problema RDP por $\lambda_{ik} \in \{0,1\}$. Sin embargo, la reformulación anterior podría presentar problemas numéricos debido a que las variables λ pueden tomar el valor cero. Introducir una simple tolerancia sobre λ para evitar dicho problema, podría incluso llevar a soluciones no factibles, (Sawaya y Grossmann, 2006). Por lo tanto una de las reformulaciones validas es la siguiente:

$$
(\lambda_{jk} + \varepsilon)(g_{jk}(\nu_{jk}/(\lambda_{jk} + \varepsilon)) + g_{jk}(0)(\lambda_{jk} - 1) \le 0
$$

Consideramos finalmente los métodos de aproximaciones exteriores y descomposición de Benders Generalizada para resolver GDPs. Como se describe en el trabajo de (Turkay y Grossmann, 1996) para valores fijos de las variables booleanas el problema resultante es el siguiente NLP:

$$\min \sum_{k\in SD} c_k + f(x)$$
$$s.a.\quad g(x)\le 0$$
$$\left.\begin{array}{l} h_{ik}(x)\le 0 \\ c_{ik}=\gamma_{ik} \end{array}\right\} \; Para\;\; Y_{i'k}=Verdad\; i'\in D,\;\; k\in SD$$
$$\left.\begin{array}{l} B^i x=0 \\ c_k=0 \end{array}\right\} \; Para\;\; Y_{ik}=Falso\; i\in D_k,\;\; i\neq i',\;\; k\in SD$$
$$x\in\Re^n,\quad c_i\in R^m$$

NLP-D

En el problema anterior (NLP-D) sólo se obliga a que se cumplan aquellas restricciones correspondientes a los términos con variables booleanas verdaderas, reduciendo la dimensionalidad del problema. Inicialmente es necesario resolver K subproblemas del tipo NLP-D, con objetivo de generar linealizaciones para cada uno de los términos en las distintas disyunciones. La selección del menor número de tales subproblemas se puede hacer resolviendo un problema de cobertura de conjunto normalmente de pequeña dimensionalidad y muy fácil de resolver (Turkay y Grossmann, 1996). Se puede entonces plantear el siguiente problema Maestro disyuntivo:

$$\min \sum_k c_k + \alpha^l$$
$$s.a.\quad \left.\begin{array}{l} \alpha \ge f(x^l) + \nabla f(x^l)(x-x^l) \\ g(x^l)+\nabla g(x^l)(x-x^l)\le 0 \end{array}\right\} \; l=1.....L$$
$$\bigvee_{i\in D_k}\begin{bmatrix} Y_{ik} \\ h_{ik}(x^l)+\nabla h_{ik}(x^l)(x-x^l)\le 0 \quad l\in L_{ik} \\ c_k=\gamma_{ik} \end{bmatrix} k\in SD$$
$$\Omega(Y)=Verdad$$
$$\alpha\in\Re,\;\; x\in\Re^n,\;\; c\in\Re^m \;\; Y\in\{Verdad, Falso\}^m$$
$$L_{ik}=\{l\,|\,Y^l_{ik}=Verdad\}$$

MILP-D

Igual que en el caso de los MINLPs en el problema anterior se pueden añadir variables de holgura para minimizar el efecto de las no convexidades.

7. OPTIMIZACIÓN GLOBAL

Una de las grandes limitaciones que aparece cuando se resuelven problemas no lineales, ya sean NLPs o MINLPs es que, en la mayor parte de las aplicaciones prácticas los problemas resultantes no son convexos. En estos casos muchos de los algoritmos presentados en los apartados anteriores sólo pueden garantizar un mínimo local al problema y en algunos casos muchas de las implementaciones no son siquiera capaces de encontrar un punto factible. En algunas aplicaciones una 'buena solución' es suficiente, pero en otras es necesario localizar el mínimo global del problema.

Los métodos de optimización global se pueden clasificar en estocásticos y deterministas. Presentaremos las líneas generales de los métodos deterministas (los métodos de enfriamiento simulado –*simulated annealing*-, algoritmos genéticos, movimiento de enjambres –*particle swarm optimization*-, etc. que se comentarán brevemente más tarde pertenecen al grupo de métodos estocásticos de optimización global) . La importancia que ha adquirido durante los últimos años la optimización global se puede valorar por el creciente número de artículos publicados en esté área. Dentro de las técnicas de optimización global se puede destacar: (a) Métodos Lipschitzian (Hansen et al, 1992a, 1992b); (b) Métodos de ramificación y acotamiento (Al-Khayyal, 1992; Al–Khayyal y Falk, 1983; Horst y Tuy, 1987); (c) Métodos de plano de corte (Tuy et al, 1985); (d) método convexo inverso (Tuy, 1987); (e) Aproximaciones Exteriores (Horst et al, 1992); (f) Métodos primal-dual (Ben-Tal et al, 1994; Floudas y Viswewaran, 1990, Shor, 1990); (g) Métodos de reformulación – linealización (Sherali y Alameddine, 1992; Sherali y Tuncbilek, 1992); y (h) Métodos de intervalo (Hansen, 1980).

Cuando en el problema aparecen estructuras especiales que causan las no convexidades (bilinealidades, términos lineales fraccionarios, términos cóncavos separables) se pueden desarrollar métodos rigurosos de optimización global. La idea es utilizar envolventes convexas o subestimadores para formular (MI)NLPs que sean problemas convexos y límites inferiores rigurosos a la solución óptima del problema original (Tawarmalani y Sahinidis, 2002). Estos estimadores se combinan con técnicas –normalmente de ramificación y acotamiento espacial- Véase (Floudas 2000; Grossmann, 1996; Quesada y Grossmann, 1995; Ryoo y Sahinidis, 1995; Zamora y Grossmann, 1999). El método de ramificación y acotamiento espacial divide la región factible de las variables continuas y compara los límites inferior y superior para eliminar subregiones, las diferencias entre los diferentes métodos están principalmente en cómo se aplica la ramificación sobre las variables continuas y discretas. Un ejemplo simple de aplicación para resolver NLPs se puede encontrar en el trabajo de (Quesada y Grossmann 1995).

Para MINLPs (Ryoo y Sahinidis. 1995), y más tarde (Tawarmalani y Sahinidis, 2004), desarrollaron un método de ramificación y acotamiento que ramifica tanto en la variables binarias como en las discretas. Dado que la calidad de los estimadores –lo cerca que están de la función a la que sustituyen- depende mucho de los límites de las variables, el algoritmo implementa una etapa de contracción de límites en cada iteración. Este algoritmo ha sido implementado en un código llamado BARON (Sahinidis, 1996) que puede utilizarse de forma autónoma o a través de algunos programas de modelado como GAMS.

En el caso de que aparezcan funciones de tipo general, continuas y diferenciables (Adjiman y Floudas, 1996) propusieron añadir un término cuadrático que si se hace lo suficientemente grande

es un estimador convexo válido de la función original.

(Smith Pantelides, 1999) propusieron la reformulación del problema junto con búsqueda espacial basándose en que cualquier problema en el que no aparezcan funciones trigonométricas puede ser reformulado en función de términos bilineales, fraccional lineales y funciones cóncavas separables. El algoritmo para resolver MINLPs ha sido implementado en gProms.

(Zamora y Grossmann, 1998; 1999) en modelos donde aparecen términos bilineales, fraccional lineales y cóncavos separables en las variables continuas y donde las variables binarias aparecen de forma lineal –nótese que cualquier problema puede ser modelado utilizando variables binarias que aparecen sólo de forma lineal- y utilizando técnicas de contracción de límites aplicaron el método de las aproximaciones exteriores en cada nodo del árbol de la búsqueda espacial.

8. MÉTODOS DE OPTIMIZACIÓN SIN DERIVADAS.

En este punto se incluyen una amplia variedad de estrategias de optimización que no necesitan derivadas. Estos métodos tienen la ventaja de que la implementación es muy sencilla y en general es necesario muy poco conocimiento previo del problema a resolver. Son especialmente adecuados para resolver problemas donde no se dispone de información acerca de las derivadas o ésta es muy imprecisa (p.e. cuando se utiliza un modelo de simulación tipo caja negra, donde la derivada es desconocida o viene afectada por ruido como consecuencia de los criterios de terminación de las ecuaciones internas).

Muchos de estos métodos se generan a través de reglas heurísticas, lo que ha dado lugar a una enorme variación de alternativas. A continuación comentaremos brevemente los más importantes.

Los métodos clásicos de búsqueda directa fueron desarrollados en los años 1960s 1970s e incluyen métodos como búsquedas en patrón, o métodos basados en diseño de experimentos (EVOP). Dentro de esta categoría caen el método de las direcciones conjugadas de (Powell, 1964), el simplex flexible de (Nedler y Mead, 1965); los métodos de búsqueda aleatoria adaptativa de (Luus y Jaakola, 1973). Todos estos métodos están basados en patrones bien definidos y se desarrollaron para problemas sin restricciones.

Sin embargo los métodos libres de derivadas más utilizados hoy en día están basados en imitar el comportamiento de algunos fenómenos naturales. Comentaremos brevemente tres de ellos: el enfriamiento simulado, los algoritmos genéticos y la optimización por movimientos de enjambres.

El enfriamiento simulado utiliza una serie de reglas heurísticas basadas en el movimiento de las moléculas cuando se produce la solidificación de un metal (Laarhoven y Aarts, 1987). El método comienza con un punto base x, y una función objetivo f(x) y un parámetro T temperatura inicial que debe elegirse. El siguiente punto x_1 es elegido al azar mediante una distribución aleatoria. Si $f(x_1) <$ f(x) el movimiento es aceptado y x_1 pasa a ser el nuevo punto inicial. En otro caso el punto x_1 no es rechazado directamente sino que se aceptará con una probabilidad p(x, x_1, T). Quizás el criterio más utilizado sea la distribución de Metropolis $p(T, x, x1) = \exp[-(f(x1) - f(x))/T]$. De esta forma aumentamos la probabilidad de localizar el óptimo global del problema aceptando movimientos que eviten quedar atrapados prematuramente en óptimos locales. A medida que avanzan las iteraciones la temperatura es reducida, disminuyendo la probabilidad de aceptar movimientos que no produzcan mejoras en la función objetivo. En el método es fundamental elegir correctamente la temperatura inicial así como la estrategia para disminuir ésta.

Los algoritmos genéticos, inicialmente desarrollados por (Holland, 1975), se basan en imitar el proceso de selección natural. Inicialmente el algoritmo selecciona de forma aleatoria una población de individuos (x_1, x_2, … , x_n), y evalúa su función objetivo. A continuación se selecciona, o bien de forma completamente aleatoria o semi-aleatoria utilizando los valores de función objetivo o algún criterio relacionado, aquellos individuos que van a ser los “padres” para la nueva generación. Los hijos se generan, a través de reglas que varían de una implementación a otra, a partir de los padres. A continuación se seleccionan los individuos que continuarán en la siguiente generación. En general un algoritmo genético utiliza tres tipos de reglas en cada iteración: 1. Reglas para seleccionar qué individuos (padres) se van a utilizar para la siguiente generación. 2. Reglas de combinación de los padres para generar los hijos. 3. Reglas para generar mutaciones y favorecer la diversidad, evitando de esta manera quedar atrapados en óptimos locales.

Los algoritmos genéticos se han hecho muy populares y hay cientos de aplicaciones diferentes. Por ejemplo, (Edgar, et al., 2002) describen un algoritmo genético disponible en MSExcel.

Los métodos basados en movimientos de enjambres de partículas (*Particle Swarm Optimization*) imitan el comportamiento de las bandadas de pájaros al volar o del movimiento de los bancos de peces. Fueron desarrollados por (Kennedy y Everhart, 1995a,b). Inicialmente el método selecciona un conjunto de puntos al azar (partículas) y evalúa el valor de la función objetivo en cada uno de esos puntos.

También inicialmente a cada partícula se le asigna una velocidad (Δx). En cada iteración el movimiento de una partícula es una combinación de su propia velocidad (inercia) y el movimiento en dos direcciones: la dirección hacia la partícula con mejor valor de función objetivo y la dirección del punto con mejor valor de función objetivo de todos los visitados por la partícula.

(Dennis y Torczon, 1991) desarrollaron una versión multidimensional que extendía el algoritmo simplex flexible de Nedler y Mead. Notaron que el algoritmo de Nedler y Mead fallaba cuando se aumentaba el número de variables incluso con problemas muy simples. Para eliminar esta dificultad combinaron las reflexiones y contracciones del método con búsquedas unidireccionales para un conjunto de direcciones linealmente independientes. Probaron que el algoritmo convergía a un óptimo local y encontraron que si se utilizaban múltiples procesadores se obtenía una sinergia inesperada que mejoraba mucho la capacidad del algoritmo. El trabajo de Dennis y Torczon, relanzó el esfuerzo investigador en el análisis y desarrollo de códigos de optimización libres de derivadas. (Conn et al., 1997) construyeron un algoritmo multivariable libre de derivadas que además incluye una región de confiabilidad. Aquí se muestrea una serie de puntos para obtener una interpolación cuadrática del modelo y por lo tanto una dirección de descenso dentro de la región de confiabilidad lo que refuerza las propiedades de convergencia. Siguiendo filosofías similares se han desarrollado códigos libres de derivadas para trabajar con modelos de optimización tipo 'caja-negra'. Estos incluyen DAKOTA (Eldred, 2002) y FOCUS desarrollado en Boeing Corporation (Booker et al, 1998).

Todos los métodos mencionados anteriormente son fácilmente aplicables a una gran variedad de problemas y son fácilmente adaptables para tratar con variables binarias (o discretas en general) si inicialmente se desarrollaron sólo para variables continuas y viceversa. Sin embargo, debido a que su criterio de terminación no está basado en información de gradiente y buscar un punto estacionario, suelen favorecer la localización de óptimos globales, no obstante no se ha descubierto todavía ninguna prueba rigurosa que permita garantizar la convergencia a un óptimo global de cualquiera de los métodos en un número finito de iteraciones.

Por otra parte, los métodos libres de derivadas funcionan bien con problemas sin restricciones o con problemas que sólo presentan límites sobre las variables, pero suelen encontrar dificultades cuando tienen que resolver problemas con restricciones. Las opciones para tratar problemas con restricciones son la eliminación de restricciones de igualdad y la utilización de funciones de penalización para las desigualdades. En general, ambos métodos han demostrado ser poco fiables. Por último, en los métodos libres de derivadas, se produce un mal escalado a medida que el número de variables aumenta (a menudo presenta crecimiento exponencial en el número de evaluaciones de la función objetivo), aunque se han conseguido resultados buenos utilizando sistemas de computación en paralelo, estos métodos raramente se pueden aplicar a problemas con más de unas pocas docenas de variables independientes.

9. SISTEMAS DE MODELADO

Podríamos considerar un lenguaje de modelado como una herramienta que permite transformar la representación matemática de un modelo a una estructura que es capaz de resolver un algoritmo de optimización en un ordenador. Aunque como se verá en los próximos párrafos un sistema de modelado hace mucho más que una simple "traducción" de un lenguaje matemático más o menos estándar a código legible por un algoritmo de ordenador.

Es fácil comprender las dificultades que aparecen cuando se plantea un modelo de optimización: El número de ecuaciones que puede aparecer en el modelo es grande, a veces enorme. Tener que 'teclear' todas esas ecuaciones es una tarea definitivamente fuera del alcance de cualquier persona y desde luego la probabilidad de cometer algún error tipográfico, enorme. Incluso aunque el modelo no sea muy grande pueden serlo los datos que lo acompañan. En muchas aplicaciones hay que calcular el valor de derivadas e incluso derivadas de segundo orden que se deben introducir también en el código. Cuando se trabaja con un modelo es a menudo interesante comprobar el funcionamiento de éste utilizando diferentes conjuntos de datos. Por supuesto, es interesante analizar los resultados, sensibilidades, etc… En más de una ocasión es necesario probar diferentes códigos disponibles, desarrollados por diferentes investigadores y que habitualmente utilizan formatos de datos distintos… De todos estos aspectos es de lo que se ocupa un sistema de modelado.

Lo primero que se debe resaltar es que la *formulación de un modelo es independiente de los formatos que tomen los códigos de resolución*. Códigos diferentes se pueden conectar a un lenguaje de modelado y es éste el que se encarga de transformar el modelo a los diferentes formatos automáticamente. Esto tiene varias ventajas, evita la etapa tediosa y con tendencia a producir errores de traducir las ecuaciones. El lenguaje de modelado detecta inconsistencias y posibles errores tipográficos y existe una clara separación entre el modelo y el algoritmo numérico de resolución, lo que tiene la ventaja de poder probar diferentes algoritmos –que pueden tener estructuras drásticamente diferentes- sólo seleccionando uno u otro.

En un lenguaje de modelado*, el modelo y los datos están separados*. De tal manera que un mismo

modelo puede ser ejecutado con diferentes conjuntos de datos sin alterar la estructura de éste. Muchos sistemas incluyen una interfase con conectividad abierta con bases de datos (ODBC *Open Database connectivity*) y una interfase con las hojas de cálculo más utilizadas.

El problema de "datos derivados" es tratado de forma automática por muchos sistemas de modelado. Especialmente con el desarrollo de la diferenciación automática (Griewank, 1991) los lenguajes de modelado generan la información derivada (gradientes, Hessianas, Jacobianos, etc) para un modelo sin que el usuario siquiera sepa que dicha importación se está generando. Los lenguajes de modelado se pueden dividir en algebraicos y no algebraicos.

9.1 Lenguajes de modelado algebraicos:

A diferencia de los lenguajes basados en procedimientos (lenguajes imperativos como C, C++, Pascal, Fortran, Java, etc; lenguajes funcionales como LISP o lenguajes de programación lógica como Prolog o ECLiPSe) donde se especifica el problema de una forma algorítmica, es decir se indica paso a paso como resolver un problema, los lenguajes de modelado algebraico almacenan el conocimiento acerca del modelo, pero no especifican como resolverlo. En este sentido son lenguajes declarativos.

Los lenguajes de modelado algebraicos son una clase especial de lenguajes declarativos, muchos de ellos diseñados para resolver problemas como el dado en la ecuación (1), o quizás alguna formulación algo más general.

Esencialmente todas las variables se convierten en unidimensionales, y el modelo se escribe en una formulación que utiliza índices de una manera muy cercana a la notación matemática habitual. Típicamente el problema se declara en forma de conjuntos (sets), índices, parámetros, y variables.
Conceptualmente, las entidades similares se agrupan en conjuntos (sets), los elementos de esos conjuntos se referencian a través de índices relacionados con cada uno de los elementos de los conjuntos. Por ejemplo la expresión $\sum_{i \in S} x_i \, y_i$ se escribiría en AMPL y GAMS respectivamente como:

```
Sum { i in S} x[i]*y[i];          (AMPL)
Eq(S(i)).. sum(i, x(i)*y(i));     (GAMS)
```

El lenguaje algebraico es el responsable de crear el problema particular, en función del modelo y los datos. Esto lo hace expandiendo la notación compacta que se consigue con los conjuntos (sets) y los índices. Comprueba que no haya errores de notación ni inconsistencias, transforma el problema específico a un formato capaz de utilizar el algoritmo deseado por el usuario, le transfiere, si es necesario la información derivada (gradientes, etc) lee los resultados, los transforma a un formato fácil de leer por el usuario (independiente del formato que utilice el algoritmo de solución) y en algunos casos analiza parcialmente la solución (análisis de sensibilidad, etc).
Los lenguajes de modelado más comunes son: AIMMS (Bisschop y Roelofs, 1999); GAMS (Brooke et al, 1992); AMPL (Fourer et al, 1993); mp-model (Ashford y Daniel, 1995); LINGO (Scharage, 1999); NOP-2 (Schichl et al, 2001); Numerica (Van Hentenryck et al, 1997); MINOPT (Schweiger et al., 1996). Una revisión excelente se puede encontrar en (Kallrath 2004).

9.2 Lenguajes de modelado no algebraicos

En ciertas áreas, especialmente donde aparecen modelos altamente estructurados, puede ser más conveniente utilizar modelos orientados a objetos.

Por ejemplo, en ingeniería química hay dos sistemas de modelado altamente especializados gPROMS (Barton y Pantelides, 1994) y ASCEND (Piela et al, 1991) que están construidos con esta filosofía. En procesos de manufactura hay un sistema llamado EXTEND (Ribera, 1998) disponible. Todos estos modelos están construidos sobre entidades básicas (reactores, tanques,…) que son ensamblados de formas distintas y con diferentes atributos. El usuario sólo tiene que especificar las partes y las conexiones y el sistema genera el modelo (total o parcialmente) de forma automática. El sistema MODELICA (Elmqvist et al, 1998) sigue está misma filosofía.

REFERENCIAS.

Adjiman, C.S. y Floudas, C.A. (1996) Rigorous convex underestimators for general twice differentiable problems. *Journal of Global Optimization,* **9**(1), 23-40.

Al-Khayyal F.A. (1992) Generalized bilinear Programming. *Mathematics of Operations Research*, **60**, 306-314.

Al-Khayyal F.A. y Falk, J.E. (1983) Jointly constrained biconvex programming. *Mathematics of Operations Research*, **8**, 273-286.

Ashford, R.W. y Daniel, R.C.; (1995). XPRESS-MP Reference Manual. Dash Associates, Blisworth House, Northants, NN73BX.

Balas, E. (1985) Disjunctive Programming and a hierarchy of relaxations for discrete optimization problems, *SIAM J.Alg. Disc. Meth*., **6**, 466-486.

Balas, E.; Ceria, S. y Cornuejols G. (1993) A lift and project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming* **58**, 295-324.

Barnhart J.R.; Johnson, E.L.; Nemhauser, G.L.; Savelsbergh, M. W. P. y Vance, P.H. (1998) Branch and Price: Column generation for solving huge integer programs. *Operations Research*, **46**. 316-329.

Barton, P. y Pantelides, C. (1994) Modeling of combined discrete/continuous processes. *AIChE Journal*. **40**, 966-979.

Beaumont, N. (1991) An Algorithm for Disjunctive Programs. *European Journal of Operations Research*, **48**, 362-371.

Ben-Tal, A.; Eiger, G.; Gershovitz, V. (1994). Global optimization by reducing the duality gap. *Mathematical Programming,* **63**,193-212.

Biegler, L.T. y Grossmann, I.E. (2004) Retrospective on Optimization. *Comp. Chem. Engng*, **28**, 1169-1192.

Binder, T.; Blank, L.; Bock, H.; Bulitsch, R.; Dahmen, W.; Diehl, M.; Kronseder, T.; Marquardt, W. Schloeder, J.; Stryk, O. (2001) Introduction to model based optimization of chemical processes on moving horizons. En Groetschel, M, Krumke, S.O. (Eds.) Online Optimization of large scale systems. Berlin: Springer.

Bisschop, J. y Roelofs, M. (1999). AIMMS: The Lenguaje Reference. *Paragon Decision Technology.* B.V., Haarlem, The Netherlands.

Bixby, R.E.; Fenelon, M.; Gu, Z.; Rothberg, E y Wunderling, R. (2002) MIP: Theory and Practice – closing the gap (http://www.ilog.com/products/optimization/tech/research/mip.pdf)

Bonami, P.; Biegler, L.T.; Conn, A.R.; Cornuéjols, G; Grossmann, I.E.; Laird, C.D.; Lee, J.; Lodi, A.; Margot, F.; Sawaya, N.; Wächter, A.; (2005) An algorithmic framework for convex mixed integer nonlinear programs. IBM Research Report RC23771 (W0511-023)

Booker, A.J.; Dennis Jr. J.E.; Frank, P.D.; Serafini, D.B.; Torczon, V.; Trosset, M.W. (1998). A rigorous framework optimization of expensive functions by surrogates. CRPC Technical Report 98739, Rice University.

Borchers, B.; Mitchell, J.E. (1994) An Improved Branch and Bound algorithm for mixed integer non linear programming. *Computers and Operations Research*. **21**, 359-367

Brooke, A.; Kendrick, D.; Meeraus, A.; (1992); GAMS: A User`s Guide. Boyd y Fraser Publising Company, Danvers, Massachusets.

Byrd, R.H.; Hribar, M.E. y Nocedal, J. (1997) An interior point algorithm for large scale nonlinear programming. Optimization technology Center, Northwestern University.

Ceria S. and J. Soares. (1999) Convex Programming for Disjunctive Optimization. *Mathematical Programming*, **86**(3), 595-614.

Conn, A.R.; Gould, N y Toint, P (2000). Trust Region Methods. Philadelphia, USA. *MPS/SIAM. Series on Optimization*

Conn, A.R.; Scheinberg, K.; Toint, P. (1997) Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, Series B **79**(3), 397.

Dakin, R.J. (1965) A tree search algorithm for mixed integer programming problems. *Computer Journal*, **8**, 250-255.

Dantzing, G.B., (1963). Linear Programming and Extensions, Princeton University Press.

Dennis, J.E. y Torczon, V. (1991) Direct search methods on parallel machines. *SIAM journal of Optics*, **1**, 448.

Duran, M.A. y Grossmann, I.E. (1986) An Outer Approximation Algorithm for a Class of Mixed Integer Non Linear Programs. *Mathematical Programming*, **36**, 307.

Edgar, T.F.; Himmelblau, D.M.; Lasdon L.S. (2002) Optimization of Chemical Processes. New York: McGrawHill.

Eldred, M. (2002). DAKOTA: A multilevel parallel object oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis.(http://endo.sandia.gov/DAKOTA/software.html).

Elmqvist, H.; Mattsson, S.; Otter, M. (1998). Modelica: the new object-oriented modeling language. http://citeseer.nj.nec.com/elmqvist98modelica.html

Fletcher, R. y Leyffer, S. (1994) Solving Mixed Integer Non Linear Programs by Outer Approximation. *Mathematical Programming* **66**, 327.

Fletcher, R. (1987) Practical methods of optimization. Chichester. Wiley.

Fletcher, R.; Gould, N. I. M.; Leyffer, S.; Toint, Ph. L. y Waechter, A (2002). Global convergence of a trust-region (SQP) filter algorithms for general nonlinear programming. *SIAM Journal on Optimization*, **13** (3) 635 - 659

Flippo, O.E.; Rinnoy Kan, A.H.G. (1993) Decomposition in General Mathematical Programming. *Mathematical Programming,* **60**, 361-382.

Floudas, C. A. y Visweswaran, V. (1990). A global optimization algorithm (GOP) for certain classes of non convex NLPs I Theory. *Computers and Chemical Engineering*, **14**, 1397-1417.

Floudas, C.A. (2000). Deterministic Global Optimization: Theory, methods and applications. Dordretcht, The Netherlands: Kluwer Academic Publishers.

Fourer, R.; Gay, D.M.; Kernighan, B.W. (1993). AMPL: A Modeling Language for Mathematical Programming. Duxbury Press, Brooks/Cole Publishing Company, Monterrey, CA.

Geofrion, A.M. (1972) Generalized Benders Decomposition. *Journal of Optimization Theory and Applications*, **10**(4), 237-260.

Griewank, A. y G Corliss, editors (1991). Automatic Differentiation of Algorithms: Theory, Implemetation and Application, Philadelphia, *SIAM*.

Grossmann, I,E. (1996). Global optimization in engineering design. Dordretcht, The Netherlands: Kluwer Academic Publishers.

Grossmann, I.E. y S. Lee. (2002) Generalized Disjunctive Programming: Nonlinear Convex Hull Relaxation and algorithms. *Computational Optimization and Applications.*

Gupta, O.K.; Ravindran, V. (1985) branch and Bound experiments in nonlinear integer programming. *Management Sciences*. **31**(12), 1533-1546.

Han, S.P. (1976) Superlinearly Convergent Variable Metric Algorithms for General Nonlinear Programming Problems. *Math Progr*., **11**, . 263-82.

Hansen, E.R. (1980). Global optimization using interval analysis: The multidimensional case. *Numerische Mathematick*. **34**, 247-270.

Hansen, P.; Jaumard, B.; Lu, S. (1992a) Global optimization of univariate Lipschitz functions. Surrey and properties. *Mathematical Programming*. **55**, 251-272.

Hansen, P.; Jaumard, B.; Lu, S. (1992b) Global optimization of univariate Lipschitz functions: New algorithms and computational comparison. *Mathematical Programming*. **55**, 273-292.

Holland J.H. (1975) Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press.

Hooker, J.N. y Osorio, M.A. (1999) Mixed logical. linear programming. *Discrete Applied Mathematics*, **96-97**, pp.395-442.

Hooker, J.N. (2000) Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction. Wiley.

Horst, R. y Tuy, H. (1987) On the convergence of global methods in multiextremal optimization. *Journal of Optimization Theory and Applications*, **54** 253.

Horst, R.; Thoai, N.V.; De Vries J. (1992). A new simplicial cover technique in constrained global optimization. *Journal of Global Optimization*, **2**, 1-19.

ILOG (2000) http://www.ilog.com/products/optimization/info/qpperformance.cfm

Johnson, E.L.; Nemhauser E. L. y Savelsbergh, M. W. P. (2000) Progress in Linear Programming based branch and bound algorithms: Exposition. *INFORMS Journal of Computing*, **12**.

Kallrath, J. (2004). Modeling languages in Mathematical Optimization. Kluwer Academic Publishers.

Karmarkar, N. (1984) A New Polynominal-time algorithm for linear programming. *Combinatorica*, **4**(4):373—395.

Kelley Jr, J.E. (1960) The Cutting Plane Method for Solving Convex Programs. *Journal of SIAM* **8**, 703-712.

Kennedy, J. y Everhart, R.C. (1995a) A discrete binary version of the particle swarm algorithm. In *Proceedings of the conference of systems, Man and Cybernetics*. 4104-4109.

Kennedy, J. y Everhart, R.C. (1995b) A new optimizer using particle swarm theory. *In proceedings of the sixth international symposium on micro machine and human science*. Nagoya Japón. IEEE service center Piscataway, NJ.

Kocis, G.R. y Grossmann. I.E. (1987) Relaxation Strategy for the Structural Optimization of Process Flowsheets. *Ind. Eng. Chem. Res*. **26**, 1869.

Kravanja, Z y Grossmann, I.E. (1994). New developments and capabilities in PROSYN an automated topology and parameter process synthesizer. *Computers Chem. Eng*. **18**, 1097-1114.

Laahorven, P.J.M y Van Aarts E.H.L. (1987). Simmulated annealing: theory and applications. Dordrecht: Reidel.

Lee, S. y I.E. Grossmann. (2000) New Algorithms for Nonlinear Generalized Disjunctive Programming. *Computers and Chemical Engineering* **24**, 2125-2141.

Leyffer, S. (2001) Integrating SQP and Branch and Bound for Mixed Integer non Linear Programming. *Computational Optimization and Applications,* **18**, 295-309.

Luus, R. y Jaakola, T.H.I. (1973) Direct search for complex systems. *AIChE Journal*, **19**, 645-646.

Ming S. Hung, Walter O. Rom, and Allan D. Waren *(1993)* Optimization with IBM OSL and Handbook for IBM OSL, *The Scientific Press (now* Duxbury Press*),*

Murtagh, B. A. y Saunders, M. A. (1987). Minos 5.1 user's guide. Technical Report SOL 83-20R. Stanford University.

Nedler, J.A. y Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, **7**, 308.

Nemhauser, G.L. y Wolsey, L.A. (1988) Integer and Combinatorial Optimization. New York Wiley Interscinece.

Nocedal, J. y Wright, S.J. (1999). Numerical Optimization. NewYork: Springer.

Piela, P., Epperly, Y.; Westerber, K; Westerberg, A. (1991) ASCEND: An object-oriented computer environment for modeling and analysis: The Modeling Language. *Computers and Chemical Engineering.* **15**(1), 53-72.

Pörn, R. y Westerlund, T. (2000) A Cutting Plane Method for Minimizing Pseudo-Convex Functions in the Mixed Integer Case. *Computers and Chemical Engineering*, **24**, 2655-2665.

Powell, M.J.D. (1964) An efficient method for finding the minimum of a function of several variables without calling derivatives. *Computing Journal*. **7**, 155.

Powell, M.J.D., (1978) A Fast Algorithm for Nonlinearly Constrained Optimization Calculations. *In Numerical Analysis*, Dundee, G.A. Watson (ed.), Lecture Notes in Mathematics 630, Springer-Verlag, Berlin.

Quesada, I. y Grossmann, I.E. (1995). A global optimization algorithm for linear fractional and bilinear programs. *Journal of Global Optimization*, **6**(1), 39-76.

Quesada, I.E. y Grossmann, I.E. (1992) An LP/NLP Based Branch and Bound Algorithm for Convex MINLP Optimization Problems. *Computers and Chemical Engineering* **16**, 937-947.

Raman, R. y I.E. Grossmann. (1991) Relation Between MILP Modeling and Logical Inference for Chemical Process Synthesis. *Computers and Chemical Engineering* **15**, 73.

Raman, R. y I.E. Grossmann. (1993). Symbolic Integration of Logic in Mixed Integer Linear Programming Techniques for Process Synthesis. *Computers and Chemical Engineering*, **17**, 909.

Raman, R. y I.E. Grossmann. (1994) Modeling and Computational Techniques for Logic Based Integer Programming. *Computers and Chemical Engineering*, **18**, 563.

Rivera, J. (1998) Modeling with EXTEND. *In Winter Simulation Conference*. 257-262.

Ryoo, H.S. y Sahinidis, N.Y. (1995) Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers and Chemical Engineering*. **19**(5), 551-566.

Sahinidis, N. V. (1996) BARON: A general purpose global optimization software package. *Journal of Global Optimization*, **8**(2), 201-205.

Sawaya, N y I.E. Grossmann (2006). Computational Implementation of nonlinear convex hull reformulation. Por aparecer en *Computers and Chemical Engineering*.

Scharage, L. (1999) Optimization Modeling with LINGO. LINDO Systems, Inc, Chicago, Il.

Schichl, H.; Neumaier, A.; Dallwig, S (2001). The NOP-2 Modeling Language. *Annals of Operations Research*, **104**, 281-312.

Schweiger, C.A.; Rojnuckarin, A.; Floudas, C.A. (1996). MINOPT: A Software Package for Mixed-Integer Nonlinear Optimization. Dept Of Chemical Engineering, Princeton University, NJ.

Sherali, H.D. y Alameddine, A. (1992). A new reformulation – linearization technique for bilinear programming problems. *Journal of Global Optimization*, **2**, 379-410.

Sherali, H.D. y Tuncbilek, C.H. (1992). A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique. *Journal of Global Optimization*, **2**, 101-112.

Shor, N.Z. (1990). Dual quadratic estimates in polynomial and Boolean programming. *Annals of Operations Research*, **25**, 163-168.

Smith, E.M.B. y Pantelides, C.C. (1999). A symbolic reformulation spatial Branch and bound algorithm for the global optimization of nonconvex MINLPs. *Computers and Chemical Engineering*, **23**, 457-478.

Stubbs R. y S. Mehrotra. (1999) A Branch-and-Cut Method for 0-1 Mixed Convex Programming. *Mathematical Programming*, **86**(3), 515-532.

Tawarmalani, M y Sahinidis, N.V. (2004). Global optimization of mixed integer non linear programs: theoretical and computational study. *Mathematical programming* Ser. A **99**, 563–591

Tawarmalani, M. y N. V. Sahinidis, (2002) Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications, Kluwer Academic Publishers, Dordrecht, Vol. **65** in *Nonconvex Optimization And Its Applications series*.

Türkay, M. and I.E. Grossmann, (1996). A Logic Based Outer-Approximation Algorithm for MINLP Optimization of Process Flowsheets. *Computers and Chemical Enginering*, **20**, 959-978.

Tuy, R. (1987). Global minimum of difference of two convex functions. *Mathematical Programming Study,* **30**, 150-182.

Tuy, R.; Thieu, T.V.; Thai, N.Q. (1985) A conical algorithm for globally minimizing a concave function over a closed convex set. *Mathematics of Operations Research*, **10**, 498-514.

Van Hentenryck, P.; Michel, J.; Deville, Y.; (1997). Numerica – A Modeling language for Global Optimization. MIT Press, Cambridge MA.

Vecchietti, A. y I.E. Grossmann. (1999) LOGMIP: A Discrete Continuous Nonlinear Optimizer. *Computers and Chemical Engineering* **23**, 555-565.

Vecchietti, A. y I.E. Grossmann. (2000) Modeling Issues and Implementation of Language for Disjunctive Programming. *Computers and Chemical Engineering* **24**, 2143-2155.

Viswanathan, J. y Grossmann, I.E. (1990) A Combined Penalty Function and Outer Approximation Method for MINLP Optimization. *Computers and Chemical Engineering,* **14**, 769.

Wächter, A. (2002) An interior point algorithm for large scale non linear optimization with applications in process engineering. Ph. D. Tesis Carnegie Mellon University.

Wätcher A. y Biegler. L.T. (2002) Global and local convergence of line search filter methods for nonlinear programming. *CAPD Technical report B-01-09*.

Westerlund, T. y Pettersson, A. (1995) A Cutting Plane Method for Solving Convenx MINLP Problems. *Computers and Chemical Engineering*, **19**, S131-S136.

Williams, H.P. (1985) Mathematical Building in Mathematical Programming. John Wiley, Chichester.

Yuan, X.; Zhang, S.; Piboleau, L.; Domenech, S. (1988) Une Methode d'optimisation Nonlineare en Variables Mixtes pour la Conception de Procedes *RAIRO* **22**, 331.

Zamora , J.M. y Grossmann, I.E. (1998) Continuous global optimization of structured process system models. *Computers and Chemical Engineering*, **22**(12), 1749-1770.

Zamora, J.M. y Grossmann, I.E. (1999) A branch and bound algorithm for problems with concave univariate. *Journal of Global Optimization*, **14**(3), 217-249.