INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Física y Matemáticas

Algoritmo Heurístico para

Optimización Mixta Multi-objetivo

T E S I S

que presenta:

Ing. Juan Esaú Trejo Espino
jesaute@hotmail.com

Para obtener el grado de
Maestro en Ciencias Fisicomatemáticas

Directora
Dra. Adriana Lara López

Ciudad de México, Junio 2016

Resumen

En este trabajo se aborda la solución de problemas de optimización lineal mixta considerando más de una función objetivo de manera simultánea. Se hace una breve revisión de los algoritmos considerados estado del arte en este tema y se propone un método heurístico original para su tratamiento. El algoritmo propuesto, denominado MOMIPGA, está compuesto por un Algoritmo Genético, diseñado particularmente para este fin, el cual ha sido hibridizado con dos técnicas tomadas de la literatura reciente: el Método Simplex Multiobjetivo[Ehrgott, 2005] y el *Triangle Splitting Method*[Boland et al., 2014].

Abstract

This work focus on the solution of mix-integer linear optimization problems when several objective functions are involved. A brief overview on state-of-the-art methods is presented, and a novel heuristic is proposed for the treatment of these problems. The proposed algorithm is called MOMIPGA and it is compound by a hybridisation of a specifically designed Genetic Algorithm, combined with tho recent new techniques from literature: the Multiobjective Simplex[Ehrgott, 2005] method and the Triangle splitting Method[Boland et al., 2014].

Agradecimientos

Mis más sinceros agradecimientos a cada una de las personas que con sus consejos, sugerencias y palabras de aliento me motivaron durante los momentos difíciles y compartieron mi alegría en momentos de éxito. Gracias a todos por siempre estar conmigo.

Además quiero expresar mi agradecimiento al Consejo Nacional de Ciencia y Tecnología por el apoyo otorgado durante mi formación académica. Asimismo, agradezco a la Dra. Adriana, quien a través de sus enseñanzas fue parte importante para realización de este trabajo.

Índice general

Introducción

Una gran cantidad de problemas de decisión y de planificación, en la vida cotidiana, implican múltiples objetivos en conflicto que deben ser optimizados de manera simultánea. Por ejemplo, en la compra de equipos de computo se busca tener un sistema de alto rendimiento, con el mínimo costo posible. En este tipo de problemas no existe una solución única, considerada la mejor con respecto a todos los objetivos a la vez. Un problema de este tipo es conocido como *problema de optimización multiobjetivo* (MOP). Los MOPs se pueden clasificar de muchas maneras, dependiendo de las características de las funciones y espacios de búsqueda que los componen. En el caso donde algunas de las variables de decisión solo pueden tomar valores enteros, el problema se denomina *problema de optimización mixta multiobjetivo* (MOMP).

Debido al gran avance en el área de optimización clásica y a la gran variedad de técnicas de solución para problemas donde se considera un solo objetivo, los métodos mas populares en las ultimas décadas para resolver MOPs se basan en una idea simple: se transforma artificialmente un MOP en un problema "equivalente" de un solo objetivo y se resuelve éste mediante métodos tradicionales de optimización. La dificultad surge porque los problemas multiobjetivo dan lugar a un conjunto de soluciones óptimas (conocido como conjunto óptimo de Pareto), en lugar de una solución puntual. Para un tomador de decisiones es valioso contar con el conjunto completo óptimo de Pareto, o al menos con la mayor cantidad de soluciones de éste. Por lo general los métodos de optimización clásica requieren ejecuciones repetidas de un algoritmo de naturaleza puntual[1] para encontrar más de una solución perteneciente al conjunto óptimo de Pareto. Desde otra visión, el enfoque poblacional[2] de métodos heurísticos como son los algoritmos genéticos, permiten una forma eficaz de encontrar simultáneamente múltiples elementos del conjunto de óptimos de Pareto, en una sola ejecución.

[1]Se conoce como puntual cuando iterativamente se va mejorando una única solución.

[2]Este término se refiere a que opera sobre conjuntos, también se utiliza el término *set-oriented*.

El problema de optimización lineal multiobjetivo considerando algunas variables enteras (llamado de *optimización mixta*) es un problema difícil, en parte porque hereda las dificultades de la Programación Entera[Bazaraa et al., 2011] y también por el hecho de ser multiobjetivo. Al igual que en muchos problemas de esta naturaleza, las técnicas de programación tradicionales y la aplicación de métodos deterministas limitan el estudio a problemas con tamaños pequeños; por lo que para el tratamiento en la práctica, las técnicas heurísticas se han vuelto las más populares. El objetivo de esta tesis es proponer una técnica de naturaleza heurística para el tratamiento de problemas de optimización lineal mixta multiobjetivo. Al ser la propuesta de naturaleza heurística, mediante un estudio experimental se busca mostrar que el método aquí propuesto produce resultados competitivos respecto al estado del arte en la materia y que además presenta ventajas en términos de su eficiencia computacional.

En el capítulo 1 se presentan los conceptos básicos de optimización multiobjetivo y se describen brevemente las técnicas más populares para resolver MOPs, esto con el fin de motivar la dificultad del problema de múltiples objetivos, en general. A continuación, en el capítulo 2 se introducen los conceptos básicos para la definición del problema de programación mixta multiobjetivo. Se revisa brevemente la teoría y los resultados principales[3] necesarios para la implementación de los métodos aquí estudiados. En el capítulo 3 se presenta el método MOMIPGA propuesto para la solución de problemas de optimización lineal mixta con dos objetivos. Dicho método se basa en un Algoritmo Genético[4] cuyos operadores son diseñados de manera particular para explotar las ventajas geométricas del problema de nuestro interés. Este algoritmo genético especializado se apoya en ciertos elementos del método Simplex Multiobjetivo[Steuer, 1989], diseñado para optimización en espacios continuos, y en el método *Triangle Splitting Method (TSM)*[Boland et al., 2014], propuesto recientemente. En el capítulo 4 se muestran los resultados experimentales obtenidos mediante la aplicación de MOMIPGA sobre una *suite* especializada de problemas de prueba aceptados en la práctica para evaluar este tipo de algoritmos. En este capítulo se muestra que, para los problemas de prueba estudiados, el algoritmo propuesto es eficaz para el cálculo de los conjuntos completos de soluciones buscadas, mostrando ser más eficiente que el TSM en cuanto al número de evaluaciones de la función objetivo requeridas, y al obtener resultados competitivos en cuanto al costo computacional

[3]En muchos de los resultados las pruebas se omiten y se refieren al trabajo original, por brevedad y al no ser dichas pruebas fundamentales para este estudio.

[4]Los algoritmos genéticos[Holland, 1975] han sido ampliamente utilizados en la práctica para aproximar soluciones a ciertos problemas que se consideran intratables con métodos exactos.

en términos del tiempo de ejecución[5]. Finalmente en el capítulo 5 se presentan las conclusiones y se menciona el trabajo a futuro.

[5]Todas las implementaciones de este trabajo se desarrollaron en los lenguajes de programación `C++` y `Matlab`. Los experimentos se llevaron a cabo en una PC con sistema operativo LINUX.

Capítulo 1

Optimización Multiobjetivo

En este primer capítulo empezaremos por mencionar algunas definiciones de optimización monobjetivo con el fin de describir y resaltar las diferencias entre las optimización clásica (una función) y la optimización multiobjetivo (2 ó 3 funciones simultaneas). Asimismo, debido a estas diferencias se mencionaran las dificultades que surgen ahora al considerar mas de un función.

Matemáticamente, se habla de optimización cuando se requiere encontrar el valor mínimo o máximo de una función que debido a la modelación, sus variables pueden estar sujetas a ciertas restricciones representadas como un sistema de igualdades y/o desigualdades. Así se entiende por optimización monobjetivo el caso donde se requiere optimizar una función escalar $f : \mathbb{R}^n \to \mathbb{R}$. El problema se expresa como

$$\begin{array}{rl} Minimizar & f(x) \\ \text{sujeto a} & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \end{array} \tag{1.1}$$

A $x = (x_1, x_2, \ldots, x_n)$ se le denomina vector de variables de decisión o simplemente variables de decisión mientras que f es la función objetivo.

Sin perdida de generalidad se puede suponer que el fin es minimizar. Cuando se requiere maximizar es suficiente con minimizar la función con signo negativo. Esto debido a que el problema de maximizar $f(x)$ es equivalente a minimizar $-f(x)$.

Definición 1.1. Un punto $x^* \in \mathbb{R}^n$ es un *mínimo* de $f : \mathbb{R}^n \to \mathbb{R}$ si $f(x^*) \leq f(x)$ para toda $x \in \mathbb{R}^n$.

Con la definición anterior se puede distinguir que la solución de un problema de optimización monobjetivo, en caso de que exista, es un único valor que puede alcanzarse en más de un punto. Se debe notar que el contra dominio de la función objetivo en este caso es un conjunto que posee un orden total y por lo tanto dados dos puntos x, $y \in \mathbb{R}^n$ es posible determinar la relación de orden que guarda su imagen sobre f. Es decir, $f(x)$ es mayor, menor ó igual que $f(y)$. Este hecho constituye una de las principales diferencias entre la optimización monobjetivo y la multiobjetivo.

En un *problema de optimización multiobjetivo* (MOP) es necesario minimizar de manera simultánea m funciones objetivo, considerando $m \geq 2$. Estas funciones se pueden expresar mediante una función vectorial $F : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$, es decir un vector con m componentes de la forma

$$F(x) = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_m(x) \end{pmatrix} \tag{1.2}$$

Es conveniente nombrar de manera particular el dominio y el contra dominio de la función; de esta manera, al espacio $\mathbb{R}^n$ se le conoce como espacio de variables o de parámetros y al espacio $\mathbb{R}^m$ espacio objetivo.

Definición 1.2. Matemáticamente el MOP está descrito como

$$\begin{aligned} \textit{Minimizar} \quad & F(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\ \text{sujeto a} \quad & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \end{aligned} \tag{1.3}$$

donde $h_i(x) \leq 0$ y $g_j(x) = 0$ representan las restricciones mencionadas, dando origen al conjunto Ω definido en la expresión (1.4) y que es conocido como el conjunto factible.

$$\Omega = \{x \in \mathbb{R}^n : h_i(x) \leq 0 \ \forall \ i = 1, 2, \ldots, p \text{ y } g_j(x) = 0 \ \forall \ j = 1, 2, \ldots, q\} \tag{1.4}$$

Una de las particularidades del MOP es la definición de optimalidad. Por ejemplo, en optimización tradicional, con una función objetivo, un punto $x^* \in \mathbb{R}^n$ se considera un mínimo de $f : \mathbb{R}^n \to \mathbb{R}$ si se cumple la siguiente condición

$$f(x^*) \leq f(x) \qquad \forall\ x \in \mathbb{R}^n$$

en cuyo caso a x^* se le denomina *vector óptimo* y $f(x^*)$ es el *valor óptimo*. Así la solución, en caso de existir, es un único valor $f(x^*)$ que aunque se puede alcanzar en varios puntos es posible distinguir que el valor óptimo de f es $f(x^*)$, pues $\mathbb{R}$ posee un orden total. En cambio, en optimización multiobjetivo el espacio objetivo, $\mathbb{R}^m$, no tiene un orden total[1] y no es posible determinar un único punto que sea mínimo. Además se presupone que las funciones objetivo se encuentran en conflicto por lo que considerar una perturbación de cierta solución $x^* \in \Omega \subset \mathbb{R}^n$ puede incrementar el valor de una función y simultáneamente causar un decremento en otra. Con la Figura 1.1 se ilustra el hecho de que las funciones se encuentren en conflicto. En la imagen se presentan dos funciones objetivo bajo el mismo dominio, sin embargo el conjunto de interés es el intervalo $[x_1^*, x_2^*]$ pues ahí se consigue un compromiso entre las funciones. Note que f_1 disminuye su valor mientras que f_2 aumenta e incluso el valor que minimiza una función causa un aumento en la otra como por ejemplo x_1^* minimiza a f_1 pero su evaluación en f_2 es mayor.

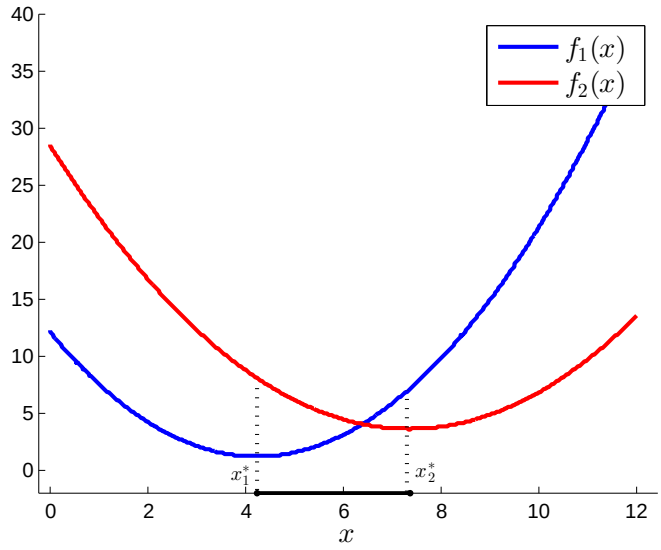

Figura 1.1: Funciones en conflicto pues el minimio de f_1 no es igual al minimo de f_2.

[1]Formalmente se puede tener un orden total considerando un ordenamiento lexicográfico, sin embargo éste impone una preferencia entre las funciones objetivo.

Para resolver los problemas multiobjetivo primero es necesario contar con una forma de decidir que soluciones se consideran óptimas. Formalmente las siguientes definiciones caracterizan los aspectos a considerar para determinar si un vector de decisión es parte de la solución de un MOP o no.

Definición 1.3. Dado un MOP con $x, y \in \Omega$ se dice que x domina a y, denotado por $x \prec y$, si y solo si se cumplen las siguientes condiciones:

1. $f_i(x) \leq f_i(y)\ \forall i \in \{1, 2, \ldots, m\}$
2. $\exists i \in \{1, 2, \ldots, m\}$ tal que $f_i(x) < f_i(y)$

Al analizar la relación de dominancia se pueden encontrar algunas propiedades interesantes que se describen a continuación:

1. Es *irreflexiva*, pues si $x \prec x$ entonces $f_i(x) < f_i(x)$ para algún $i \in \{1, 2, \ldots, m\}$ lo cual es falso.
2. Es *asimétrica*, dado que $x \prec y$ no implica que $y \prec x$. De hecho, si x domina a y entonces y no domina a x.
3. La relación es asimétrica y por lo tanto no es antisimétrica. Para ser antisimétrica se debe cumplir que dado que $x \prec y$ y $y \prec x$ entonces $x = y$.
4. Siguiendo la Definición 1.3 se puede notar que si $x \prec y$ y $y \prec z$ entonces $x \prec z$, por lo que es *transitiva*.

Con estas propiedades podemos notar que la relación de dominancia define un orden parcial estricto pues es irreflexiva, asimétrica y transitiva. Este hecho ha sido usado en estudios de aspecto teórico en MOPs [Ehrgott, 2005]. A través de la relación de dominancia podemos evaluar los puntos en Ω que no son dominados y encontrar el conjunto de puntos compromiso que se desea. Para esto nos basamos en las siguientes definiciones de optimalidad de Pareto [Pareto, 1971].

Definición 1.4. Un punto $x^* \in \Omega$ es *óptimo de Pareto* si y sólo si $\nexists\ x \in \Omega$ tal que $x \prec x^*$.

Definición 1.5. Un punto $x^* \in \Omega$ es un punto *óptimo débil de Pareto* si y sólo si $\nexists\ x \in \Omega$ tal que $f_i(x) < f_i(x^*)$ para $i \in \{1, \ldots, m\}$.

Definición 1.6. Se denomina *conjunto de óptimo de Pareto*, o *conjunto de Pareto*al conjunto $\mathcal{P}_s$ formado por todos los puntos que son óptimos de Pareto, es decir:

$$\mathcal{P}_s = \{x \in \Omega : x \text{ es óptimo de Pareto}\}$$

Asimismo, a la imagen del conjunto de Pareto se le llama *frente de Pareto* y esta definido por

$$\mathcal{P}_f = \{F(x) \in \mathbb{R}^m : x \in \mathcal{P}_s\}$$

.

Con las definiciones anteriores se puede notar que la solución del MOP está formada por los conjuntos $\mathcal{P}_s$ y $\mathcal{P}_f$, notando que $\mathcal{P}_s \subset \mathbb{R}^n$ y $\mathcal{P}_f \subset \mathbb{R}^m$. Es así como la solución de un MOP esta constituida por un conjunto de puntos, por lo cual el objetivo de cualquier algoritmo computacional para resolver MOPs es devolver los conjuntos $\mathcal{P}_s$ y $\mathcal{P}_f$ ó una aproximación discreta representativa de éstos. Para tener una idea mas clara, en la Figura 1.2 se considera como espacio objetivo $\mathbb{R}^2$ y se muestran una serie de puntos que son la imagen de puntos dominados, óptimos débiles y óptimos de Pareto. Por ejemplo, los puntos F^1 y F^{10} son óptimos débiles de Pareto, mientras que F^5 está dominado por F^4 y F^7 domina a F^8. Además el frente de Pareto, de este conjunto discreto, esta conformado como $\mathcal{P}_f = \{F^2, F^3, F^4, F^6, F^7, F^9\}$.

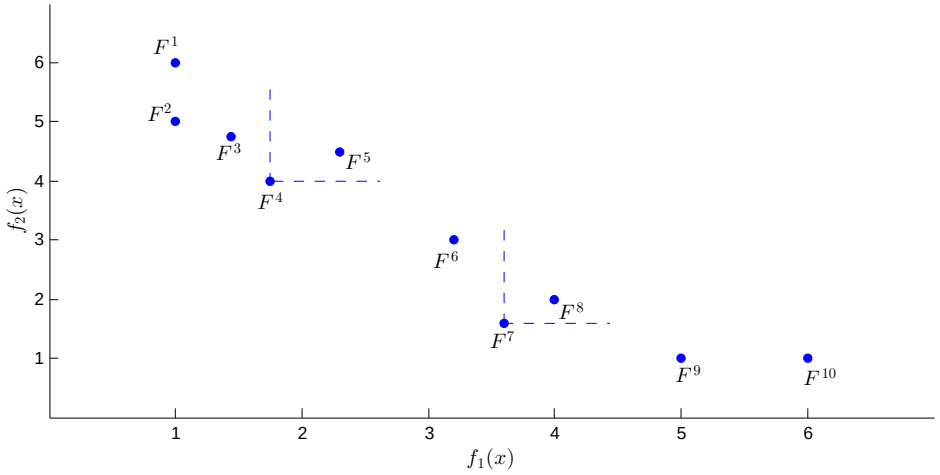

Figura 1.2: Ejemplo de puntos dominados y óptimos débiles de Pareto en el espacio objetivo.

Finalmente, la siguiente definición nos ayudará a comprender mas adelante la forma en que un algoritmo poblacional, como el *algoritmo genético* (AG), puede inicializar las soluciones.

Definición 1.7. Sean $f_1^*, \ldots, f_m^*$ los valores óptimos para cada función objetivo, respectivamente. Se denomina *hiperplano utópico* al hiperplano en $\mathbb{R}^m$ que comprende los puntos $f_1^*, \ldots, f_m^*$.

1.1. Métodos tradicionales de solución

Como se ha descrito en la sección anterior, la solución de un MOP está conformada por un conjunto de puntos en el dominio del problema. La primera idea intuitiva que surgió para calcular dicho conjunto fue transformar el problema original en un problema de optimización clásico donde sólo hay una función objetivo. Esta idea resulta interesante porque en la actualidad existe una amplia variedad de métodos para optimizar dichas funciones ($f : \mathbb{R}^n \to \mathbb{R}$) [Nocedal and Wright, 2006] [Luenberger and Ye, 2008]. Sin embargo el uso de estos métodos nos lleva a encontrar una solución puntual por cada transformación del problema original. Si se requiere encontrar todo el conjunto de puntos óptimos de Pareto, es necesario aplicar varias veces tales métodos. En lo siguiente se presentan tres técnicas que a través de esta idea obtienen soluciones de un MOP.

1.1.1. Método lexicográfico

El método lexicográfico se puede considerar como un método *a priori* para encontrar soluciones de un MOP. Inicialmente este método se usó para generar soluciones óptimas de Pareto a partir de soluciones óptimas débiles. Al igual que otros métodos, éste se basa en transformar el problema original en un problema de optimización escalar o mono-objetivo. La idea principal es resolver m subproblemas considerando sólo una función objetivo a la vez. La secuencia que se sigue para resolver cada subproblema depende de la prioridad que se asigne a cada función objetivo. Esta prioridad se conoce como *orden lexicográfico* y está definida por el tomador de decisiones. La solución buscada se obtiene de optimizar las funciones objetivo en secuencia siguiendo el orden lexicográfico definido. Sin embargo en cada subproblema se usa el valor óptimo de las funciones obtenido en el subproblema anterior. Tal valor se usa para asegurar que el valor de las funciones que ya se optimizaron se preserve.

Para un MOP como el descrito en (1.3), supongamos que el orden lexicográfico definido para los objetivos coincide con su subíndice. Así por ejemplo, f_i tiene prioridad i entre los objetivos. Entonces el método lexicográfico inicia resolviendo el problema monobjetivo para f_1. Las funciones objetivos restantes se omiten, como se muestra en (1.5).

$$\begin{array}{ll} \text{mín} & f_1(x) \\ \text{s. a} & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \end{array} \tag{1.5}$$

Al resolver el primer subproblema se obtiene el valor óptimo de $f_1(x)$, el cual denotaremos por f_1^*. Como se mencionó antes, este valor se usa como referencia para mantener el valor óptimo de f_1 ó en el mejor de los casos encontrar el óptimo global en caso de aún no alcanzarlo. Esto se logra a través de la restricción $f_1(x) \leq f_1^*$. Así para el subproblema siguiente se optimizará f_2 y se agregará la restricción $f_1(x) \leq f_1^*$, como se ve en (1.6).

$$\begin{array}{ll} \text{mín} & f_2(x) \\ \text{s. a} & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \\ & f_1(x) \leq f_1^* \end{array} \tag{1.6}$$

Este proceso se sigue hasta llegar al último subproblema donde se optimiza la $m-$ésima función considerando los $m-1$ valores óptimos de las funciones anteriores. En general, el $k-$ésimo subproblema está descrito por (1.7).

$$\begin{array}{ll} \text{mín} & f_k(x) \\ \text{s. a} & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \\ & f_1(x) \leq f_1^* \\ & f_2(x) \leq f_2^* \\ & \quad \vdots \\ & f_{k-1}(x) \leq f_{k-1}^* \end{array} \tag{1.7}$$

La solución obtenida al resolver el último subproblema se toma como la solución al problema multiobjetivo. En algunas ocasiones se usa la expresión (1.8) para denotar a la solución de un problema multiobjetivo con un orden lexicográfico dado.

$$\text{lex mín}\{f_1, f_2, \ldots, f_k : g_i(x) \leq 0 \ \forall i = 1, \ldots \text{ y } h_j(x) = 0 \ \forall j = 1, \ldots q\} \tag{1.8}$$

Es claro entonces que al aplicar este método solo se obtiene un punto del Frente de Pareto. Cuando es necesario conocer más de un punto del frente se puede modificar el orden lexicográfico y aplicar de nuevo el método.El siguiente ejemplo muestra su uso para un problema multiobjetivo con dos funciones objetivo.

Ejemplo 1.1.1. Solución de un problema multiobjetivo (problema de Belegundu), donde $F(x) : \Omega \subseteq \mathbb{R}^2 \to \mathbb{R}^2$, usando el método lexicográfico.

$$\begin{aligned} \text{mín} \quad & F(x) = \begin{pmatrix} f_1(x) = -2x_1 + x_2 \\ f_2(x) = 2x_1 + x_2 \end{pmatrix} \\ \text{s. a} \quad & \\ & -x_1 + x_2 - 1 \leq 0 \\ & x_1 + x_2 - 7 \leq 0 \\ & 0 \leq x_1 \leq 5 \\ & 0 \leq x_2 \leq 3 \end{aligned}$$

Consideremos que el orden lexicográfico esta definido por el subíndice de las funciones objetivo. En dicho caso empezamos por resolver el problema para minimizar $f_1(x)$, omitiendo por ahora la segunda función objetivo. En la Figura 1.3 se puede apreciar el conjunto factible Ω del problema. Al aplicar métodos de optimización monobjetivo obtenemos como solución el punto $x^* = (5, 0)$, con el cual el valor mínimo de $f_1(x)$ es $f_1^* = f_1(x^*) = -10$. Ahora usamos este valor para restringir el valor de $f_1(x)$ y minimizar $f_2(x)$. El subproblema final se muestra en (1.9).

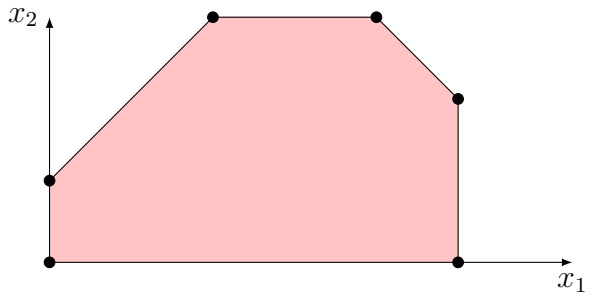

Figura 1.3: Region factible para el problema del ejemplo 1.1.1

$$\begin{array}{ll} \text{mín} & f_2(x) = 2x_1 + x_2 \\ \text{s. a} & \\ & -x_1 + x_2 - 1 \leq 0 \\ & x_1 + x_2 - 7 \leq 0 \\ & -2x_1 + x_2 \leq -10 \\ & 0 \leq x_1 \leq 5 \\ & 0 \leq x_2 \leq 3 \end{array} \tag{1.9}$$

De nuevo empleamos métodos de optimización monobjetivo para resolver (1.9). La solución que se obtiene es $x^* = (5, 0)$ cuyo valor en la función objetivo es $f_2^* = f_2(x^*) = 10$. La solución final para el problema multiobjetivo es $x^* = (5, 0)$ con un valor $F(x^*) = (-10, 10)^T$. Para seguir con el ejemplo, supongamos que se desea conocer otro punto del frente. Para obtenerlo invertimos el orden lexicográfico. Así minimizamos primero $f_2(x)$ y después usamos su valor óptimo en el subproblema siguiente, donde hemos de minimizar $f_1(x)$. La solución para el problema multiobjetivo es $x^* = (0, 0)$ con $F(x^*) = (0, 0)^T$. En la Figura 1.4 se pueden observar las dos soluciones que se obtuvieron del método lexicográfico y el frente de Pareto real para este problema.

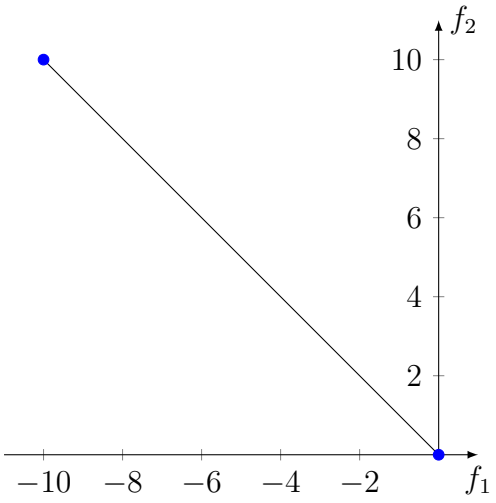

Figura 1.4: Frente de Pareto del ejemplo 1.1.1

Una ejecución del método lexicográfico sólo devuelve un punto del frente de Pareto. La solución depende del orden definido en las funciones objetivo por lo que para

cada orden distinto se puede encontrar una solución diferente. En particular todas las soluciones encontradas son puntos extremos del frente de Pareto. Para mostrar este hecho se analiza el segundo subproblema generado, el cual se muestra en (1.6). Se puede notar que la restricción que se agrega modifica el conjunto factible Ω, dejando únicamente el punto x_1^* como factible. Entonces el mínimo de $f_2(x)$ sujeto a $x \in \{x_1^*\}$ es $f_2^* = f_2(x_1^*)$. Al final la solución obtenida es x_1^*. Sin embargo, ésta se consigue al minimizar únicamente $f_1(x)$ con $x \in \Omega$. El uso de este método es adecuado cuando hay soluciones dominadas en los extremos del frente de Pareto. Suponga que se tiene el siguiente problema

$$\begin{aligned} \text{mín} \quad & F(x) = \begin{pmatrix} 3x_1 - 2x_2 + 10 \\ x_1 + 2x_1 - 3x_2 + 18 \end{pmatrix} \\ \text{s. a} \quad & \\ & x_1 + 2x_2 + x_3 \leq 10 \\ & 2x_1 - x_2 - x_3 \leq 8 \\ & -x_1 + x_2 - x_3 \leq 6 \\ & -2x_1 - x_2 - x_3 \leq -3 \\ & 1 \leq x_1 \leq 5 \\ & 0 \leq x_2 \leq 3 \\ & 0 \leq x_3 \leq 5 \end{aligned} \tag{1.10}$$

Usando el método lexicográfico, minimizamos primero $f_1(x) = 3x_1 - 2x_2 + 10$. El mínimo es $f_1^* = 7$ con $x^* = (1, 3, 0)$. Notemos que $F(x^*) = (7, 25)^T$. Ahora agregamos la restricción $3x_1 - 2x_2 + 3 \leq 0$ para minimizar $f_2(x) = x_1 + 2x_2 - 3x_3 + 18$. La solución final es $x^* = (1, 3, 3)$ con $F(x^*) = (7, 16)^T$. En la Figura 1.5 se muestra en frente de Pareto para este problema. La linea punteada representa todos los puntos débilmente dominados por $F(x^*) = (7, 16)^T$. Al inicio se obtuvo el punto $F(x^*) = (7, 25)^T$. Sin embargo es posible determinar una mejor solución para f_2 sin alterar el valor de f_1. Claramente el método ayudó a descartar todos estos puntos débilmente dominados que se encontraban en un extremo del frente de Pareto.

Discusión

Una clara ventaja de este método es el uso de métodos de optimización monobjetivo. Además es útil cuando el tomador de decisiones debe considerar ciertas funciones como prioridad debido a limitaciones del modelo. Sin embargo, cuando no esta clara la secuencia que se seguirá la elección se hace arbitrariamente. Por otra parte, Stadler argumenta que las restricciones que se agregan pueden reemplazarse por igualdades [Stadler, 1988].

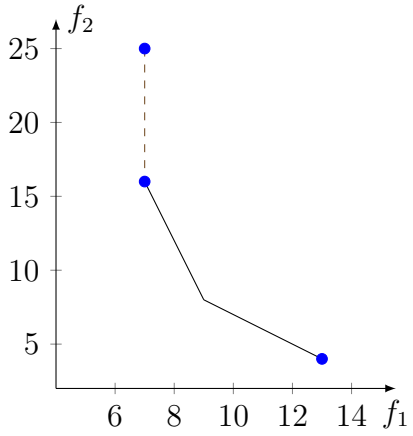

Figura 1.5: Frente de Pareto para el problema descrito en 1.10

La principal desventaja es que devuelve únicamente un punto del frente de Pareto. Además si en los óptimos independientes de cada función no hay puntos débilmente dominados entonces la solución que se obtiene es el punto óptimo de cada función. Así este método puede estar limitado sólo a los problemas donde se encuentran puntos débilmente dominados.

Para contrarresta la desventaja de encontrar sólo los puntos óptimos de cada función se han propuesto dos variantes [Marler and Arora, 2004] muy similares entre sí. La idea es relajar cada desigualdad que se va agregando al problema durante el proceso. En la primer variante se define una valor de variación para cada restricción que se agrega [Waltz, 1967]. El $k-$ésimo subproblema queda descrito por (1.11), donde δ_r con $r = 1, 2, \ldots, k-1$ es una valor de variación para los óptimos da cada función. Cada δ_r se establece por el tomador de decisiones y debe ser un valor pequeño respecto al valor óptimo de la $r-$ésima función objetivo.

$$\begin{aligned} \text{mín} \quad & f_k(x) \\ \text{s. a} \quad & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \\ & f_1(x) \leq f_1^* + \delta_1 \\ & f_2(x) \leq f_2^* + \delta_2 \\ & \quad \vdots \\ & f_{k-1}(x) \leq f_{k-1}^* + \delta_{k-1} \end{aligned} \tag{1.11}$$

Por otra parte en [Osyczka, 1984] se propone agregar las restricciones en cada etapa de la forma (1.2). En este caso δ_r representa el porcentaje que se esta dispuesto a relajar para el valor óptimo. Por ello, los valores para cada δ_r deben variar entre 0 y 100. Uno puede variar δ_r de distintas formas para de esta manera generar diferentes puntos óptimos de Pareto.

$$f_r(x) \leq \left(1 + \frac{\delta_r}{100}\right) f_r^* \qquad r = 1, \ldots, k-1 \tag{1.12}$$

Cabe destacar que a pesar de relajar las restricciones, se supone que las variaciones son pequeñas respecto a los óptimos obtenidos de resolver problemas previos y no sobre los óptimos de cada función objetivo. Por lo que el método obtiene puntos del frente de Pareto cerca de los óptimos independiente de cada función y no sobre la región conocida comúnmente como *rodilla*.

1.1.2. Método de la restricción ϵ

Este método, conocido como $\epsilon-$constraint, fue propuesto por primera vez en 1971 [Haimes et al., 1971]. El método modifica el MOP para tratarlo como un problema monobjetivo. El usuario debe establecer una preferencia sobre alguna de las funciones objetivo, la cual se minimizará. Para cada una de las funciones restantes se define una cota superior que se introduce al modelo en forma de restricción. El problema MOP a través del método $\epsilon-$constraint se visualiza en (1.13), donde se minimiza la $k-$ésima función objetivo. En la tercer linea de las restricciones se muestran las demás funciones restringidas a cierta cota ϵ_r que el usuario define.

$$\begin{aligned} &\text{mín} \quad f_k(x) \\ &\text{s. a} \\ &\qquad g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ &\qquad h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \\ &\qquad f_r(x) \leq \epsilon_r \text{ para } r = 1, 2, \ldots, m \;\; \& \;\; r \neq k \end{aligned} \tag{1.13}$$

Cuando es necesario obtener más de un punto del frente de Pareto el usuario puede establecer distintas cotas para cada función. Sin embargo, esto es un inconveniente, pues el usuario tiene que elegir los límites superiores ϵ_r adecuados. Para ello, se debe optimizar cada función objetivo de manera independiente ya que su valor óptimo da idea del rango en el que se puede establecer ϵ_r. En la Figura 1.6 se muestra un problema con dos funciones objetivo donde se establecen cinco distintas cotas para $f_1(x)$. La cota ϵ_1^a muestra que la restricción $f_1(x) \leq \epsilon_1^a$ es infactible y por lo tanto no hay solución alguna. Por otra parte, la cota ϵ_1^e es tan amplia que el método determina

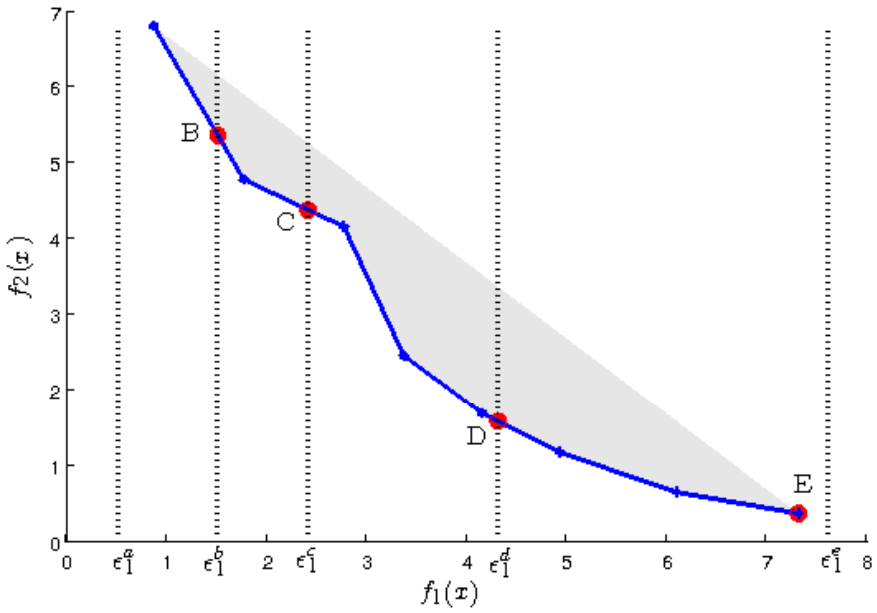

Figura 1.6: Espacio objetivo para un problema biobjetivo con distintos valores para ϵ_1

el extremo donde $f_2(x)$ alcanza su mínimo valor. Además cuando se ocupa la cota ϵ_1^c se ve que el método genera soluciones en regiones no convexas.

Discusión

Algunas ventajas del método $\epsilon-$constraint son:

- Se aplican métodos de optimización mono-objetivo.
- Útil cuando en el modelo se prefiere una función o sólo se requiere una parte del frente.
- No toma en cuenta la convexidad del frente de Pareto. Genera puntos donde otros métodos no son capaces de hacerlo.

Sin embargo existen algunas desventajas como por ejemplo que para aplicar el método $\epsilon-$constraint es necesario conocer los óptimos de cada función, lo que significa resolver m problemas de optimización. La aplicación de este método sólo devuelve

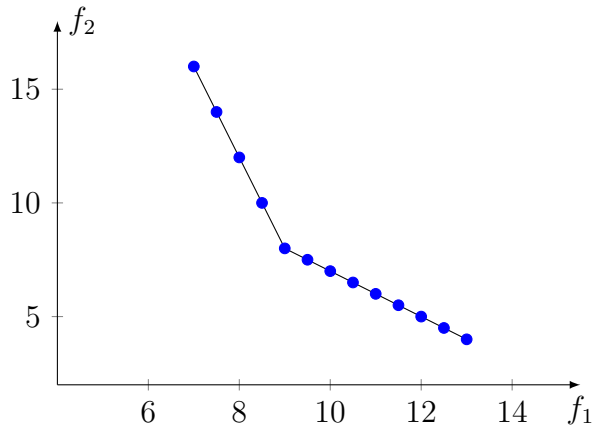

Figura 1.7: Frente de Pareto para el problema descrito en 1.10. Cada punto corresponde a una solución obtenida por el método $\epsilon-$constraint.

un punto del Frente de Pareto, para generar varias soluciones es necesario variar el vector ϵ. En la Figura 1.7 se muestra el frente de Pareto (linea continua) del problema 1.10. Además se pueden visualizar ciertos puntos, los cuales corresponden una solución usando el método de la restricción ϵ con un vector de cotas ϵ distinto.

Este método puede parecer idéntico a las variantes del método lexicográfico, sin embargo hay que recordar que éstos no son capaces de generar la rodilla del frente pues las cotas que se definen a cada función están limitadas a una pequeña variación del óptimo de la función en cuestión. Además en el método de la restricción ϵ no se conocen los óptimos de cada función, por lo que las cotas establecidas son de cierta manera arbitrarias y están definidas por el tomador de decisiones de acuerdo a su experiencia.

1.1.3. Método de la suma ponderada

El método de suma ponderada o suma de pesos ha sido uno de los más utilizados para resolver MOPs, pues se tienen resultados teóricos que garantizan obtener puntos óptimos de Pareto bajo ciertas circunstancias. Como su nombre lo indica, la idea es asociar a cada función objetivo una ponderación o coeficiente de peso y minimizar la suma de estos objetivos. Entonces considerando $\alpha \in \mathbb{R}^m$ como el *vector de pesos* tal que $\alpha_k \geq 0 \ \forall \ k = 1, \ldots, m$ y $\sum_{k=1}^{m} \alpha_k = 1 \ \Rightarrow \|\alpha\|_1 = 1$ se tiene que el problema de escalarización asociado es

$$\begin{aligned} \text{mín} \quad & \sum_{k=1}^{m} \alpha_k f_k(x) \\ \text{s. a} \quad & \\ & g_i(x) \leq 0 \text{ para } i = 1, 2, \ldots, p \\ & h_j(x) = 0 \text{ para } j = 1, 2, \ldots, q \end{aligned} \tag{1.14}$$

Notemos que (1.14) es un problema de optimización escalar donde se minimiza la combinación convexa de las funciones objetivo. En principio, entre más ponderaciones distintas se consideren para resolver el problema más puntos de $\mathcal{P}_s$ se obtendrán.

Teorema 1.1. *Sea x^* una solución a* (1.14) *para un vector de pesos dado. Si x^* es única entonces, es un punto óptimo de Pareto.*

Demostración. Ver [Miettinen, 1998], pág. 79.

Teorema 1.2. *La solución del problema* (1.14) *es óptimo de Pareto si el vector de pesos α es tal que sus componentes son estrictamente positivas, es decir, $\alpha_k > 0$ para $k = 1, \ldots, m$.*

Demostración. Ver [Miettinen, 1998], pág. 78.

Discusión

Los teoremas anteriores dan garantía de obtener un punto óptimo de Pareto del MOP considerando sólo problemas de optimización escalar. El problema multi-objetivo se reduce a varios problemas escalares; tantos como vectores de peso se determinen, y se obtiene para cada uno un punto del frente de Pareto.

Sin embargo, el funcionamiento de este método se basa en la convexidad del frente de Pareto. Para problemas con un frente no convexo el método de suma de pesos es incapaz de generar las regiones no convexas [Das and Dennis, 1997], más aún en caso de no tener alguna región convexa el método no devuelve más que los puntos mínimos de cada función objetivo independientemente. Esta es una gran desventaja ya que no se conoce *a priori* la convexidad del frente de Pareto. Por otra parte, para un vector de pesos α dado la solución de (1.14) no necesariamente es única.

Capítulo 2

Optimización Lineal Multiobjetivo

Los MOPs se pueden clasificar de distintas maneras, que pueden depender de las características del problema en cuestión o no. En una primera clasificación, incluida en este capítulo, se presentan problemas donde tanto las funciones como las restricciones son lineales, y se encuentran definidas sobre dominios continuos, para después introducir a los problemas de *programación lineal mixta multiobjetivo* (MOMIP), donde las funciones y restricciones son lineales, pero además algunas variables sólo toman valores enteros. Entonces, un *problema de optimización lineal multiobjetivo* (MOLP) es un caso de problema multiobjetivo donde se considera que las funciones y las restricciones son lineales. En tal caso el problema se puede reescribir en forma matricial como en (2.1), donde $C \in \mathbb{R}^{m\times n}$ es la matriz de las funciones objetivo, $A \in \mathbb{R}^{l\times n}$ es la matriz de restricciones y $b \in \mathbb{R}^l$ un vector. Además el conjunto factible está definido por $\Omega = \{x \in \mathbb{R}^n : Ax \leq b\}$

$$\begin{array}{ll} \text{mín} & F(x) = Cx \\ \text{s. a} & \\ & Ax \leq b \end{array} \tag{2.1}$$

Por su relación con la programación lineal, en ocasiones los MOLPs también se conocen como problemas de programación lineal multiobjetivo. En general, es claro que la función objetivo de un MOLP, $F : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}^m$ se puede ver como una transformación lineal cuya matriz asociada es C. Pues notemos que si $x, y \in \Omega$ y $\alpha \in \mathbb{R}$

$$F(x+y) = C(x+y) = Cx + Cy = F(x) + F(y)$$

$$F(\alpha x) = C(\alpha x) = \alpha(Cx) = \alpha F(x)$$

Además el conjunto factible Ω define un poliedro, pues resulta ser la intersección de l semiespacios definidos por cada una de las desigualdades de la matriz A y el vector b. Por otra lado la matriz C se puede expresar de forma tal que cada fila corresponde a los coeficientes de cada función objetivo, es decir

$$F(x) = Cx = \begin{pmatrix} C_1 \\ \vdots \\ C_m \end{pmatrix} x$$

Teorema 2.1. *[Ehrgott, 2005] Sea $\Omega \in \mathbb{R}^n$ el conjunto factible de un MOLP y $\Upsilon = \{F(x) \in R^m : x \in \Omega\}$ el conjunto imagen del conjunto factible. Entonces Ω y Υ son convexos y cerrados.*

Definición 2.1. Un punto $x \in \Omega$ es un *punto extremo* de Ω si y sólo si $\nexists\ x^1, x^2 \in \Omega$ con $x^1 \neq x^2$ tal que $x = \lambda x^1 + (1-\lambda)x^2$ para toda $\lambda \in [0,1]$.

Un punto que pertenece al conjunto factible es punto extremo o vértice si no es combinación convexa de dos elementos distintos del conjunto. En otras palabras, un punto extremo es un punto que no es un punto interior de algún segmento de línea totalmente contenida en Ω.

Definición 2.2. Se dice que d es una *dirección* de Ω si para todo $x \in \Omega$ el conjunto $R = \{x + \lambda d : \lambda \in [0, +\infty)\}$ está contenido en Ω.

Definición 2.3. Dado el conjunto factible Ω, se dice que d es una *dirección extrema* si y sólo si $\nexists\ d^1, d^2 \in \Omega$ con $d^1 \neq d^2$ tal que $d = \lambda d^1 + (1-\lambda)d^2$ para toda $\lambda \in [0,1]$.

Un conjunto poliédrico es un caso especial de conjunto convexo. Un conjunto poliédrico es la intersección de un número finito de semiespacios cerrados. El Teorema 2.2 menciona que cualquier elemento de un conjunto poliédrico se puede expresar como combinación lineal de sus puntos y direcciones extremas.

Teorema 2.2. *[Bazaraa et al., 2011] Sea Ω un conjunto poliédrico no vacío y sea $x \in \Omega$. Sean $x^1, x^2, \ldots, x^k$ y $d^1, d^2, \ldots, d^l$ los puntos extremos y direcciones extremas de Ω, respectivamente. Entonces existen $\lambda_i \in [0,1]$ para $i = 1, 2, \ldots, k$ y $\beta_j \in [0, \infty)$ para $j = 1, 2, \ldots, l$ con $\sum_{i=1}^{k} \lambda_i = 1$ tal que*

$$x = \sum_{i=1}^{k} \lambda_i x^i + \sum_{j=1}^{l} \beta_j d^j$$

Definición 2.4. Un conjunto $\Omega \subseteq \mathbb{R}^n$ es acotado si $\exists r > 0$ y $\omega \in \mathbb{R}^n$ tales que $\Omega \subseteq B(\omega, r)$.

Cuando un conjunto poliédrico es acotado se dice que es un poliedro. En este caso un poliedro no posee direcciones extremas y por lo tanto cualquier elemento del conjunto sólo se puede expresar a partir de sus puntos extremos, como lo menciona siguiente teorema:

Teorema 2.3. *[Bazaraa et al., 2011] Sea Ω un poliedro, $x \in \Omega$. Sean $x^1, x^2, \ldots, x^k$ puntos extremos de Ω. Entonces existen $\lambda_i \in [0,1]$ para $i = 1, 2, \ldots, k$ con $\sum_{i=1}^{k} \lambda_i = 1$ tal que*

$$x = \sum_{i=1}^{k} \lambda_i x^i$$

.

Teorema 2.4. *[Ehrgott, 2005] Sea $\Upsilon = \{F(x) \in R^m : x \in \Omega\}$ el conjunto imagen del conjunto factible. Si $\Upsilon \neq \emptyset$ y existe un $y \in \mathbb{R}^m$ tal que $\Upsilon \subset y + \mathbb{R}^m_{\geq}$ entonces $\mathcal{P}_f \neq \emptyset$.*

Teorema 2.5. *[Ehrgott, 2005] Sea Ω el conjunto factible. Si Ω es acotado entonces $\mathcal{P}_s$ y $\mathcal{P}_f$ son conjuntos conectados.*

Se puede notar a partir de estos resultados que el conjunto factible es además un poliedro, por lo que resulta claro el uso del método Simplex para resolver este tipo de problemas. Sin embargo, en este caso tenemos dos o más funciones objetivo. Para considerar ambas funciones se realizan algunas modificaciones al método Simplex obteniendo un algoritmo capaz de encontrar el frente de Pareto exacto de un MOLP.

2.1. Método para espacios continuos

A partir de esta sección nos limitaremos, por simplicidad, a explicar el caso particular de dos funciones objetivo. El método Simplex parametrizado o método biSimplex es un método iterativo que resuelve MOLPs con dos objetivos ($m = 2$), el cual se puede entender como una extensión del método Simplex usado para problemas de optimización lineal monobjetivo. Para comprender el sentido de esta extensión primero se hace notar que un problema donde existen dos funciones objetivo se puede expresar como un problema escalar, al parametrizar las funciones, como en el caso del método explicado en la sección 1.1.3. Para ello se introduce el paramétro λ y se define la función objetivo parametrizada como una combinación convexa de C_1 y C_2 como en la ecuación (2.2), donde se debe cumplir que $0 \leq \lambda \leq 1$.

$$C_\lambda = \lambda C_1 + (1-\lambda)C_2 \tag{2.2}$$

2.1.1. Elementos básicos del método Simplex

El método Simplex es un procedimiento muy eficiente para resolver problemas de optimización lineal mediante el uso de una computadora. Como hemos analizado antes, el conjunto factible de un MOLP define un conjunto poliédrico, el cual posee puntos extremos. El método Simplex encuentra el punto óptimo, donde cierta función objetivo alcanza su mínimo, moviéndose a lo largo de los vértices y evaluando cada punto extremo. Esto basado en el hecho de que el punto que minimiza la función objetivo es un punto extremo del conjunto factible, como se mostrará a continuación.

Los problemas de optimización lineal están definidos en forma matricial como en (2.1). Sin embargo en los problemas monobjetivo, C es un vector fila de dimensión n, es decir, $C \in \mathbb{R}^{1\times n}$. Entonces consideremos el siguiente problema

$$\begin{array}{ll} \text{mín} & Cx \\ s.\ a & \\ & Ax \leq b \\ & 0 \leq x. \end{array} \tag{2.3}$$

Sean $p_1, p_2, \ldots, p_k$ los puntos extremos del conjunto factible y sea x cualquier punto tal que $Ax = b$ y $0 \leq x$, por el Teorema 2.3 tenemos que x puede expresarse como:

$$x = \sum_{j=1}^{k} \alpha_j p_j$$

donde $\alpha_j \geq 0$, para $j = 1, \ldots, k$ y $\sum_{j=1}^{k} \alpha_j = 1$. Así el problema (2.3) se puede expresar en función de las variables $\alpha_1, \ldots, \alpha_k$ como:

$$\begin{array}{ll} \text{mín} & \sum_{j=1}^{k} (Cp_j)\alpha_j \\ s.\ a & \\ & \sum_{j=1}^{k} \alpha_j = 1 \\ & \alpha_j \geq 0 \end{array} \tag{2.4}$$

Para resolver este problema podemos simplemente encontrar el índice correspondiente al valor mínimo de todos los Cx_j's, es decir

$$i = \underset{j}{argmin} \{Cp_j \text{ para } j = 1, \ldots, k\}$$

y considerar $\alpha_i = 1$ y todas las demás iguales a cero. Así se encuentra el valor óptimo de la función objetivo, sin embargo notemos que este valor es la evaluación del punto extremo p_i sobre la función objetivo, $f(p_i) = Cp_i$. Por lo que se puede concluir que el valor óptimo de $f(x)$ se encuentra en alguno de los puntos extremos del conjunto factible.

Notemos además que si el mínimo Cx_j ocurre en más de un índice, entonces cada punto extremo correspondiente es un punto óptimo e incluso cada combinación convexa de estos puntos también es una solución óptima.

El método Simplex encuentra una solución óptima con una secuencia finita de pasos sobre el sistema de ecuaciones del problema original. Para ello se eligen m variables, que se denominan dependientes o básicas, pues éstas se expresan en términos de las variables restantes; variables que se denominan independientes o no básicas. Si a cada variable no básica le asignamos el valor cero se dice que la solución encontrada es una solución básica. Si además cada variable básica es no negativa entonces la solución es básica factible. Consideremos el sistema $Ax = b$ como en el problema (2.3), con $x \geq 0$ y supongamos que el rango de A es m. Entonces podemos definir la base como la matriz B que consta de un arreglo de m columnas de A. En este sentido la matriz B se denomina *matriz básica* o simplemente *base* debido a que los índices de las columnas de A que la forman corresponde a los índices de las variables básicas. Las columnas restantes de A forman la matriz no básica, la cual se denota por N.

El método Simplex inicia con un sistema de ecuaciones formado por las restricciones (en forma canónica) con respecto al conjunto de variables básicas. Por ejemplo, supongamos que tenemos el siguiente sistema de ecuaciones con las variables básicas $x_1, x_2, \ldots, x_m$. Notemos que el sistema (2.5) es equivalente al sistema (2.3), pues si se puede obtener un sistema de igualdades a través de sumar a cada restricción una variable no negativa, éstas variables comúnmente se denominan *variables de holgura*. Así $\bar{A}$ denota la matriz aumenta y en un abuso de notación x es el vector de variables de decisión junto con las variables de holgura.

$$\begin{array}{ccccccccc} -z & & & & +\bar{c}_{m+1}x_{m+1} & + \ldots + \bar{c}_n x_n & = & -\bar{z}_0 \\ & x_1 & & & +\bar{A}_{1,m+1}x_{m+1} & + \ldots + \bar{A}_{1,n}x_n & = & \bar{b}_1 \\ & & x_2 & & +\bar{A}_{2,m+1}x_{m+1} & + \ldots + \bar{A}_{2,n}x_n & = & \bar{b}_2 \\ & & & \ddots & \vdots & \vdots & = & \vdots \\ & & & x_m & +\bar{A}_{m,m+1}x_{m+1} & + \ldots + \bar{A}_{m,n}x_n & = & \bar{b}_m \end{array} \tag{2.5}$$

El objetivo es encontrar el valor de $x_1, \ldots, x_n$ y minimizar z tal que satisfacen (2.5), donde $\bar{A}_{i,j}$, $\bar{c}_j$, $\bar{b}_i$ y $\bar{z}_0$ son constantes. Los coeficientes $\bar{c}_j$ de la función objetivo se denominan costos relativos debido a que su valor es relativo o depende de la elección de las variables básicas. En notación matricial este sistema se puede reescribir como en (2.6).

$$\begin{aligned} -z + 0x_B + \bar{c}^T x_N &= -\bar{z}_0 \\ Ix_B + \bar{A}x_N &= \bar{b} \end{aligned} \tag{2.6}$$

De la expresión anterior podemos observar que el valor de la función objetivo para la base correspondiente es

$$z = \bar{z}_0 + \sum_{j \in \mathcal{N}} \bar{c}_j x_j \tag{2.7}$$

donde $\mathcal{N}$ es el conjunto de índices de las variables no básicas. Notemos que si para algún índice s el costo relativo es negativo, es decir supongamos que $\bar{c}_s < 0$; dado que nuestro objetivo es encontrar el mínimo de la función objetivo, se torna conveniente incrementar el valor de x_s tanto como sea conveniente, para obtener una reducción en el valor de z. Sin embargo como x_s es hasta ahora una variable no básica, las variables básicas dependen de ésta y por lo tanto no puede crecer de manera arbitraria, ya que puede alterar la factibilidad de las soluciones. Entonces de la expresión (2.6) tenemos que

$$Ix_B = \bar{b} - \bar{A}_{*,s}x_s$$

donde $\bar{A}_{*,s}$ denota la columna s de la matriz $\bar{A}$. Cuando todas las entradas de $\bar{A}_{*,s}$ son negativas entonces x_s puede incrementar su valor sin limite, en tal caso se concluye que el problema de optimización no está acotado. Por lo que solo nos enfocaremos en las entradas positivas. Así, si elegimos la entrada r tal que

$$r = \underset{j}{argmin} \left\{ \frac{\bar{b}_j}{\bar{A}_{j,s}}, \text{donde } \bar{A}_{j,s} > 0 \right\} \tag{2.8}$$

y hacemos que $x_s = \dfrac{\bar{b}_r}{\bar{A}_{rs}}$ entonces la variable básica $x_r = \bar{b}_r - \bar{A}_{r,s}x_s = 0$, por lo que ésta deja de ser una variable básica y es reemplazada en el algoritmo por x_s. En conclusión la variable x_s se vuelve variable básica y x_r es ahora una variable no básica.

Regresando a la expresión (2.7), cuando todos los costos relativos son positivos no hay manera de mejorar el valor de z y en tal caso el procedimiento se termina; concluyendo que el óptimo se ha alcanzado.

2.1.2. Método Multisimplex

En esta sección se explicara la versión del método Simplex usada cuando se minimiza simultáneamente dos funciones objetivo, en este caso el método Simplex Multiobjetivo es un método iterativo que resuelve *problema de optimización lineal biobjetivo*s (BOLPs) (por lo que en ocasiones también es llamado Simplex Biobjetivo), como el descrito en la ecuación (2.1). Usando el hecho del Teorema 2.1 y del método de suma de pesos sabemos que encontrar soluciones no dominadas de (2.1) es equivalente a resolver el problema (2.9)

$$\begin{array}{ll} \text{mín} & [\lambda C_1 + (1-\lambda)C_2]\, x \\ s.\ a & \\ & Ax \leq b \\ & 0 \leq x. \end{array} \tag{2.9}$$

Es claro que (2.9) es un problema de optimización mono-objetivo que depende del paramétro λ, pues utiliza la función parametrizada (2.2). Entonces, supongamos que tenemos el siguiente problema de optimización biobjetivo donde se requiere encontrar el frente de Pareto.

Ejemplo 2.1.1.

$$\begin{array}{ll} \text{mín} & F(x) = \begin{pmatrix} 1.5x_1 - 2x_2 \\ -3x_1 + x_2 \end{pmatrix} \\ \text{s. a} & \\ & -x_1 + 4x_2 \leq 15 \\ & x_1 + 1.5x_2 \leq 12.5 \\ & 4x_1 - 0.5x_2 \leq 24 \\ & -x_1 - 4x_2 \leq -6 \\ & -3x_1 - x_2 \leq -7 \end{array} \tag{2.10}$$

La función parametrizada para este ejemplo queda definida como

$$F_\lambda(x) = C_\lambda x = (4.5\lambda - 3)x_1 + (-3\lambda + 1)x_2 \tag{2.11}$$

En la Figura 2.1 podemos observar la dirección sobre la cual cada función objetivo decrece de manera independiente, tal dirección corresponde al negativo del gradiente de cada función. En este caso, dado que las funciones son lineales el gradiente de cada función corresponde con su vector de costos. Por ejemplo, para $f_1(x) = 1.5x_1 - 2x_2$, $\nabla f_1(x) = (1.5, -2) = C_1$. En la Figura también se observa el poliedro derivado de las restricciones, el cual define al conjunto factible.

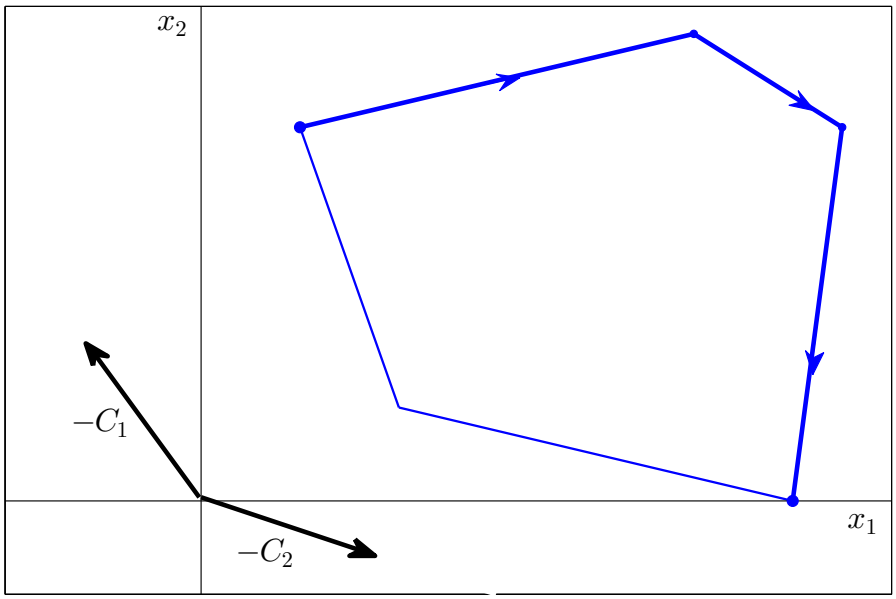

Figura 2.1: Espacio de las variables del Ejemplo 2.1.1

Si aplicamos el método gráfico a la función parametrizada podemos encontrar las soluciones que forman el conjunto de Pareto. Observemos que $-C_1$ y $-C_2$ delimitan un cono de direcciones sobre las cuales la función parametrizada puede encontrar su mínimo. Si hacemos variar el parámetro λ de 0 a 1 y para cada valor se encuentra el mínimo de la función parametrizada se obtendrá una secuencia de puntos que pertenecen al frente de Pareto. Sin embargo, como se mostró en la sección 2.1.1, un punto que es óptimo es además un punto extremo del poliedro, por lo que no es necesario

realizar la búsqueda variando λ sobre todo el intervalo $(0, 1)$, pues como se observa en la Figura 2.2 para todo un intervalo de valores de λ la solución que se obtiene corresponde al mismo punto extremo. Así para $\lambda \in [0.73, 1]$ la solución óptima es el punto $A = (1, 4)$, para $\lambda \in [0.56, 0.73]$ la solución corresponde al punto $B = (5, 5)$, para $\lambda \in [0.25, 0.56]$ la solución es $C = (6.5, 4)$ y para $\lambda \in [0, 0.25]$ es $D = (6, 0)$.

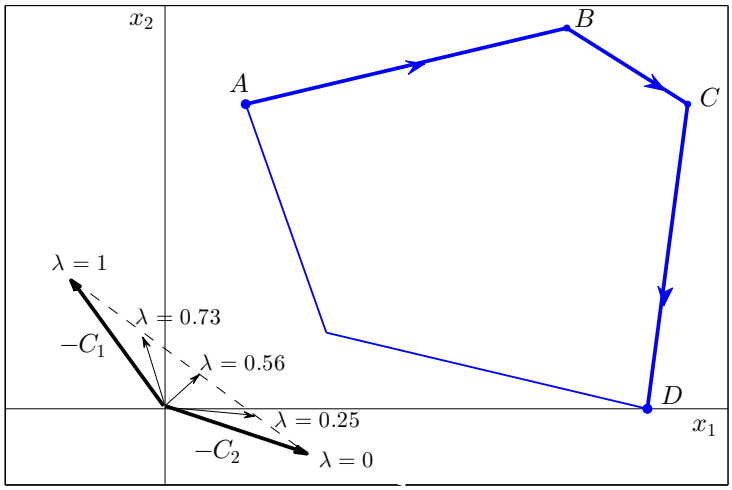

Figura 2.2: Espacio de las variables de decisión correspondiente al Ejemplo 2.1.1

Este ejemplo ilustra de manera sencilla los detalles del método bisimplex, los cuales se presentan a continuación. Una característica de los MOLPs que es consecuencia de los teoremas 2.1 y 2.3 es el hecho de que los puntos extremos del frente de Pareto son la imagen de ciertos puntos extremos del conjunto factible en el espacio de las variables. En la Figura 2.3 se puede apreciar el método gráfico usando un valor de λ dentro de los intervalos definidos antes. Notemos por ejemplo que para obtener el punto B, λ puede variar entre 0.56 y 0.73, por lo que $\lambda = 0{,}7$ es el valor que se eligió para generar la imagen y como se observa en efecto se obtiene el punto B.

Dado que la imagen del conjunto factible es un conjunto convexo, podemos encontrar sus puntos extremos y así construir el frente de Pareto a partir de ellos. Por ello nuestro interés es determinar los valores de λ que son relevantes para encontrar tales puntos. Así para el problema (2.9) se tiene la siguiente función objetivo parametrizada, que para el método Simplex es análoga a la expresión en (2.7).

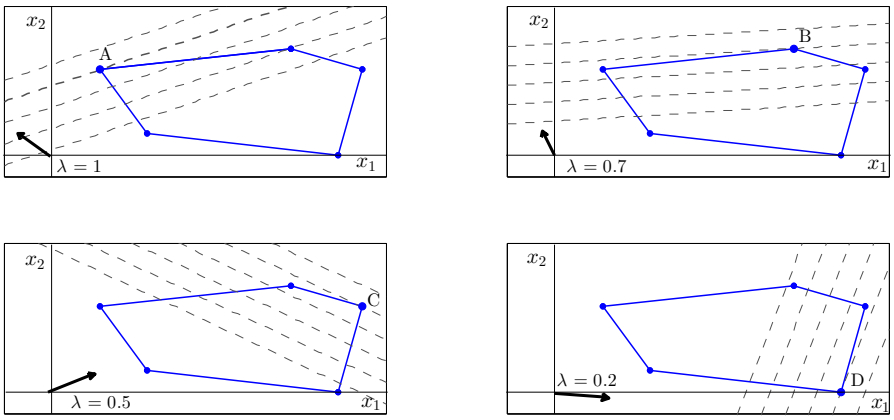

Figura 2.3: Método gráfico para cierto λ de cada intervalo critico. Solución al Ejemplo 2.1.1.

$$z = \bar{z}_0 + \sum_{j \in \mathcal{N}} \left[\lambda \bar{c}_j^1 + (1-\lambda)\bar{c}_j^2\right] x_j \tag{2.12}$$

Supongamos que hemos encontrado una base factible óptima $\hat{\mathrm{B}}$ para este problema cuando $\lambda = \hat{\lambda}$. Por el criterio de optimalidad se tiene que

$$\hat{\lambda}\bar{c}_j^1 + (1-\hat{\lambda})\bar{c}_j^2 \geq 0 \quad \forall \quad j \in \mathcal{N} \tag{2.13}$$

A continuación podemos distinguir entre dos casos para $\bar{c}_j^2$. El primero cuando $\bar{c}_j^2 \geq 0$ para toda $j \in \mathcal{N}$, en tal caso se puede mostrar que para $\lambda < \hat{\lambda}$ se cumple la ecuación (2.14), sin importar el signo de $\bar{c}_j^1$. Por lo tanto $\hat{\mathrm{B}}$ sigue siendo una base óptima para $0 \leq \lambda \leq \hat{\lambda}$.

$$\lambda\bar{c}_j^1 + (1-\lambda)\bar{c}_j^2 \geq 0 \quad \forall \quad j \in \mathcal{N} \tag{2.14}$$

Por otra parte, si existe algún $j \in \mathcal{N}$ tal que $\bar{c}_j^2 < 0$ entonces existe un valor $\lambda < \hat{\lambda}$ para el cual $\lambda\bar{c}_j^1 + (1-\lambda)\bar{c}_j^2 = 0$. Al desarrollar esta ecuación podemos calcular el valor de dicho λ como se expresa en la ecuación (2.15).

$$\lambda = \frac{-\bar{c}_j^2}{\bar{c}_j^1 - \bar{c}_j^2} \tag{2.15}$$

Se define el conjunto $I = \{j \in \mathcal{N} : \bar{c}_i^1 \geq 0, \bar{c}_i^2 < 0\}$ y se calculan los posibles valores de λ para cada índice $i \in \mathrm{I}$. Si además el máximo de ellos esta definido por λ^*, es decir

$$\lambda^* = \max_{i \in \mathrm{I}} \left\{ \frac{-\bar{c}_i^2}{\bar{c}_i^1 - \bar{c}_i^2} \right\} \tag{2.16}$$

se puede notar que $\hat{\mathrm{B}}$ es una base óptima del problema (2.9) para toda $\lambda \in \left[\lambda^*, \hat{\lambda}\right]$.

De la ecuación (2.12) tenemos que, cuando $\lambda = \lambda^*$

$$\lambda^* \bar{c}_i^1 + (1 - \lambda^*)\bar{c}_i^2 = 0$$

por lo que podemos encontrar una nueva base óptima al introducir la variable x_s en la base, donde el índice $s = i$ está definido por el máximo de los valores de los λ's posibles de la ecuación (2.16). Para elegir la variable x_r que sale de la base simplemente calculamos el índice r como en la expresión (2.8) del método Simplex.

Tomando todo el procedimiento anterior se puede resolver un BOLP usando la forma parametrizada (2.9). Primero resolviendo con $\lambda = 1$ y de manera iterativa encontrar los valores críticos λ^* y modificar la base cambiando las variables básicas de acuerdo con (2.8) y (2.16) hasta que $\bar{c}_j^2 \geq 0$ para toda $j \in \mathcal{N}$. En cada iteración se puede obtener el valor de λ^* correspondiente y el punto extremo óptimo del espacio de las variables como su valor imagen en el espacio de las funciones. El pseudocódigo para el caso de dos objetivos se muestra en el Algoritmo 2.1.1.

Para comprender la aplicación de este algoritmo estudiaremos el siguiente ejemplo.

Ejemplo 2.1.2. Problema de Ehrgott.

$$\begin{aligned} \text{mín} \quad & F(x) = \begin{pmatrix} 3x_1 + x_2 \\ -x_1 - 2x_2 \end{pmatrix} \\ \text{s. a} \quad & \\ & x_2 \leq 3 \\ & 3x_1 - x_2 \leq 6 \end{aligned} \tag{2.17}$$

Primero escribimos este problema en forma estándar cambiando las restricciones de desigualdad por igualdades. Para ello, introducimos nuevas variables, que en este tipo de problemas se denominan *variables de holgura*. Entonces el problema original es equivalente al definido en (2.18).

Algoritmo 2.1.1 Descripción del algoritmo Simplex para optimización lineal biobjetivo

Entrada: Datos de optimización lineal biobjetivo. Matrices C, A y el vector b.
Salida: Secuencia de puntos extremos óptimos en el Frente de Pareto

Sea $\mathcal{B}$ el conjunto de índices básicos y $\mathcal{N}$ el conjunto de índices no básicos.
1: Resolver el problema de optimización usando C_λ, con $\lambda = 1$

2: **while** $I = \{i \in \mathcal{N} : \bar{c}_i^1 \geq 0, \bar{c}_i^2 < 0\} \neq \emptyset$ **do**
3: $\quad s = argmax\ \{i \in I : \dfrac{\bar{c}_i^2}{\bar{c}_i^2 - \bar{c}_i^1}\}$
4: $\quad r = argmin\ \{j \in \mathcal{B} : \dfrac{\bar{b}_j}{\bar{A}_{sj}} \text{ con } \bar{A}_{sj} > 0\}$
5: $\quad$ Actualizar $\mathcal{B}$, $\mathcal{B} = (\mathcal{B} \setminus \{r\}) \cup \{s\}$
6: $\quad$ Actualizar $\mathcal{N}$, $\mathcal{N} = (\mathcal{N} \setminus \{s\}) \cup \{r\}$
7: **end while**

$$\begin{aligned} \text{mín} \quad & F(x) = (3x_1 + x_2, -x_1 - 2x_2)^T \\ \text{s. a} \quad & \\ & x_2 + x_3 = 3 \\ & 3x_1 - x_2 + x_4 = 6 \end{aligned} \tag{2.18}$$

En este problema notemos que $C_1 = (3, 1)$, $C_2 = (-1, -2)$ y por lo tanto $C_\lambda x = \lambda(3x_1 + x_2) + (1 - \lambda)(-x_1 - 2x_2)$. Como paso inicial se tiene que resolver usando como función objetivo C_λ y tomando $\lambda = 1$. Equivalente a resolver usando solamente $f_1(x) = C_1 x = 3x_1 + x_2$.

Como ahora se ha reducido a un problema monobjetivo podemos usar el método Simplex en su formato de tableu. Así el tableu óptimo es

	x_1	x_2	x_3	x_4	
z	3	1	0	0	0
x_3	0	1	1	0	3
x_4	3	-1	0	1	6

Tabla 2.1: Tableu para el Ejemplo 2.17.

Ahora al ingresar la segunda función objetivo y haciendo las adecuaciones en la base obtenemos el siguiente tableu. Éste muestra que nuestra primer punto óptimo es $x = (0, 0)$, el cual corresponde al punto $F(x) = (0, 0)$ en el frente de Pareto.

	x_1	x_2	x_3	x_4	
z_1	3	1	0	0	0
z_2	-1	-2	0	0	0
x_3	0	1	1	0	3
x_4	3	-1	0	1	6

Tabla 2.2: Tableu para el Ejemplo 2.17.

A partir de este punto podemos iniciar nuestro procedimiento con el método Simplex Multiobjetivo. En este caso buscamos el índice de las variables no básicas que forman al conjunto $I = \{i \in \mathcal{N} : \bar{c}_i^1 \geq 0, \bar{c}_i^2 < 0\}$, en nuestro caso $I = \{1, 2\}$. Ahora calculemos

$$\frac{\bar{c}_1^2}{\bar{c}_1^2 - \bar{c}_1^1} = \frac{1}{4}$$

$$\frac{\bar{c}_2^2}{\bar{c}_2^2 - \bar{c}_2^1} = \frac{2}{3}$$

con este paso podemos conocer el índice s tal que $\frac{\bar{c}_s^2}{\bar{c}_s^2 - \bar{c}_s^1}$ es máximo, en nuestro ejemplo $s = 2$. Por lo cual la variable no básica que pasara a formar parte de la base es x_2. Siguiendo con el procedimiento ahora debemos analizar el posible incremento de x_2 para que no afecte a las demás variables de la base. Al igual que en el Simplex monobjetivo debemos encontrar r tal que

$$r = argmin\{j \in \mathcal{B} : \frac{\bar{b}_j}{\bar{A}_{2j}} \text{ con } \bar{A}_{2j} > 0\}.$$

Para nuestro ejemplo la única posibilidad es $r = 3$. Por lo que la variable x_3 saldrá de la base. En el tableu hacemos los cálculos correspondientes y obtenemos la Tabla 2.3. En este caso, la solución encontrada es $x = (0, 3)$ con el punto en el frente de Pareto $F(x) = (3, -6)$. En lo siguiente repetimos el procedimiento, seleccionando los índices s y r que determinaran los cambios en el tableu. Cada solución que encontramos corresponde a un punto extremo que forma el frente de Pareto. Para

	x_1	x_2	x_3	x_4	
z_1	3	0	-1	0	-3
z_2	-1	0	2	0	6
x_2	0	1	1	0	3
x_4	3	0	1	1	9

Tabla 2.3: Tableu para el Ejemplo 2.17.

nuestro ejemplo la ultima solución encontrada se muestra en la Tabla 2.4. Note que esta última solución es equivalente a resolver el problema usando C_λ con $\lambda = 0$. En la Figura 2.4 se muestra a la izquierda el conjunto de óptimos de Pareto mientras que la imagen de la derecha corresponde al frente de Pareto. Como se aprecia en todo el proceso anterior, el método bisimplex devuelve los puntos extremos del poliedro que son solución. Sin embargo, como resultado de los teoremas 2.1 y 2.5 podemos asegurar que la combinación convexa de dos puntos subsiguientes es también parte del conjunto de Pareto y en consecuencia su imagen sobre F será parte del frente de Pareto.

	x_1	x_2	x_3	x_4	
z_1	0	0	-2	-1	-12
z_2	0	0	$\frac{7}{3}$	$\frac{1}{3}$	9
x_2	0	1	1	0	3
x_4	1	0	$\frac{1}{3}$	$\frac{1}{3}$	3

Tabla 2.4: Tableu para el Ejemplo 2.17.

Como se puede apreciar el método bisimplex es una técnica de gran utilidad para resolver MOLPs sin importar el número de variables del problema. Además como está basado en el método Simplex de programación lineal hereda las mismas propiedades de éste, como por ejemplo la garantía de obtener una solución en tiempo polinomial. En la siguiente sección estudiaremos un caso especial de los MOLPs. En este caso algunas variables se restringen a tomar valores enteros por lo que el método Simplex Multiobjetivo deja de ser una método viable para resolver el problema. Por ello se estudian otros métodos especializados en este caso especial de MOLP, sin embargo este caso resulta ser tan difícil de resolver que se propone el uso de heurísticas.

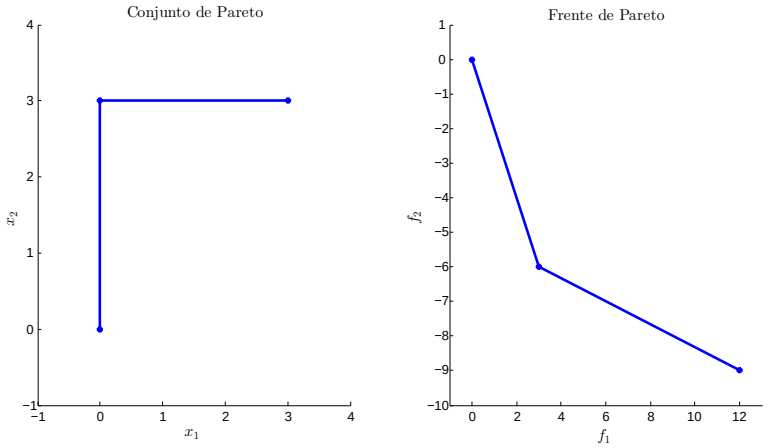

Figura 2.4: Correspondientes $\mathcal{P}_s$ y $\mathcal{P}_f$ para el Ejemplo 2.17

2.2. Optimización lineal mixta multiobjetivo

Desde la década de 1980, se han estudiado y desarrollado métodos para problemas multiobjetivo con funciones y restricciones lineales (MOLPs) cuyas variables son todas continuas [Steuer, 1989]. Sin embargo, existen varias áreas de aplicación donde es necesario considerar variables discretas para incorporar fenómenos discretos dentro del modelo matemático. Por ejemplo en [Demirtas and Özden Üstün, 2008], los autores describen el problema de seleccionar distintos proveedores al momento de realizar una orden de compra. Tal problema consiste en determinar las cantidades de compra de distintos productos (variables continuas) y el proveedor que los surtirá (variables discretas binarias, 1 si se elije el proveedor ó 0 si no). En este artículo, los autores consideran tres funciones objetivo: minimizar tanto el costo total como la tasa de defectos y maximizar el valor total de la compra, medido en calidad, cantidad y costos. Asimismo en [Mavrotas and Diakoulaki, 2005a] se trata un problema de dos objetivos: minimizar el costo de producción y las emisiones de CO_2 durante la expansión de un sistema de generación de energía eléctrica en la isla griega Creta.

Este tipo de problemas en los que hay variables de decisión tanto discretas como continuas y varios objetivos en conflicto son parte de la *programación lineal mixta multiobjetivo* (MOMIP). Matemáticamente éstos están descritos por

$$\min_{x\in\Omega} \quad F(x) = (f_1(x), \ldots, f_k(x))^T \tag{2.19}$$

donde ahora el conjunto factible esta definido por $\Omega = \{x \in \mathbb{Z}^{n_e} \times \mathbb{R}^{n_c} : Ax \leq b\}$.

Notemos que las restricciones del problema siguen siendo lineales, por lo que el problema se representa en forma matricial. En este caso n_e y n_c representan el número de variables enteras y continuas, respectivamente.

La introducción de variables discretas en problemas de optimización lineal multiobjetivo hace a estos problemas mucho más difíciles de abordar, pues conlleva las dificultades de la optimización multiobjetivo y la optimización mixta. Aunque el problema sea lineal, el conjunto factible ya no es convexo. Además, en la mayoría de los casos, los problemas no pueden ser manejados por adaptaciones monobjetivo del problema pues la forma del frente de Pareto tiene ciertas características que la hacen peculiar. Dado que el problema es lineal se pueden encontrar regiones donde la linea que une dos puntos no dominados pertenece a $\mathcal{P}_f$ pero además dadas las restricciones de integridad (valores enteros de las variables) también puede haber regiones que solo consideran un punto. En la Figura 2.5 se muestra un ejemplo del frente de Pareto de un MOMIP.

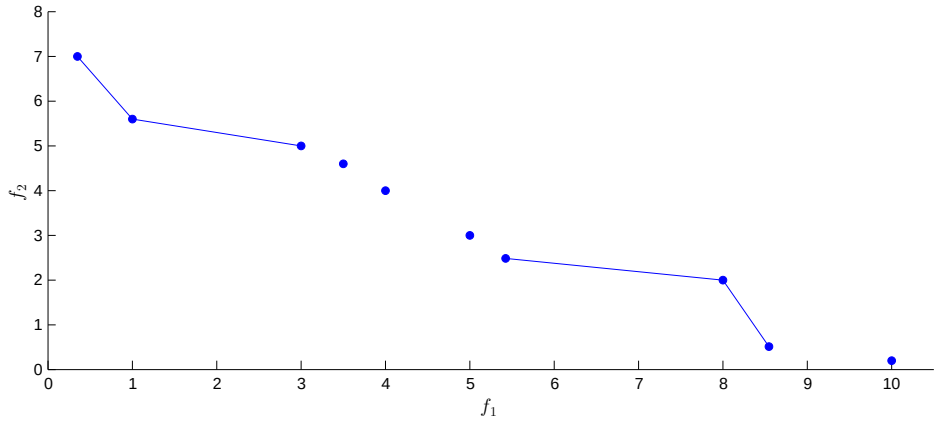

Figura 2.5: Ejemplo del $\mathcal{P}_f$ de un BOMIP.

Por otra parte, existen enfoques multiobjetivo diseñados para problemas con todas las variables enteras que no pueden aplicarse al caso mixto. Por lo tanto, las técnicas para hacer frente a MOMIP implican más que la combinación de enfoques de MOLP con técnicas de optimización mixta.

Una de las formas más comunes de obtener una aproximación de $\mathcal{P}_s$ y $\mathcal{P}_f$ es a través del método de suma ponderada. Éste consiste en minimizar una combinación convexa de los objetivos, estableciendo un vector de pesos λ cuyas entradas son

no negativas y su suma es igual a 1. Así para cada distinto vector λ se espera obtener un punto de Pareto. Sin embargo, a diferencia de MOLP, no todos los puntos óptimos de Pareto del problema mixto se pueden obtener con tal método. Dado que la región factible es no convexa, pueden existir óptimos de Pareto que no se obtienen por minimizar una combinación convexa de los objetivos. En particular, tenemos el siguiente ejemplo cuyo frente de Pareto se muestra en la Figura 2.6. Asimismo se pueden apreciar los puntos obtenidos al usar el método de suma ponderada. Nótese que éstos no corresponden a todas las regiones del frente sino únicamente a puntos extremos de algunas rectas por lo que en el mejor de los casos se obtendrán los dos extremos que definen la recta; como en la parte inferior del frente de Pareto mostrado en la imagen.

Ejemplo 2.2.1.

$$
\begin{aligned}
\text{mín} \quad & F(x) = \begin{pmatrix} 8x_1 - 3x_2 - 5x_3 \\ -6x_1 + 3x_2 + 2x_3 \end{pmatrix} \\
\text{s. a} \quad & \\
& 16x_1 - 21x_2 - 17x_3 \leq 10 \\
& 21x_1 - 11x_2 + 24x_3 \leq 230 \\
& -19x_1 - 2x_2 + 23x_3 \leq 75 \\
& 21x_1 - 23x_2 - 1x_3 \leq 200 \\
& 7x_1 - 24x_2 + 15x_3 \leq 200 \\
& 8x_1 + 12x_2 + 17x_3 \leq 156 \\
& 8x_1 - 3x_2 - 5x_3 \leq -31 \\
& x_1,\ x_2 \in \mathbb{Z}
\end{aligned}
\tag{2.20}
$$

Este hecho ha llevado a los investigadores del área a distinguir entre las soluciones obtenidas por una suma ponderada y las que no, lo que da lugar a la siguiente definición.

Definición 2.5. Sea $x^* \in \mathcal{P}_s$ y $\lambda \in \mathbb{R}^m$ tal que $\lambda_1, \ldots, \lambda_m \geq 0$, $\sum_{i=1}^{m} \lambda_i = 1$. Si $x^* = \text{mín}_{x \in \Omega} \lambda^T F(x)$, entonces x^* se denomina un *óptimo de Pareto soportado* y $F(x^*)$ un *punto no dominado soportado*.

De nuevo, la Figura 2.6 nos ayudará a explicar por qué algunos puntos no se pueden obtener al minimizar una combinación convexa de las funciones objetivo. Enfoquemos nuestra atención en la parte central del frente de Pareto y notemos que la combinación convexa del punto A y B; recta que une tales puntos, domina la región central del frente. Por ello, cuando usa el método de suma ponderada minimizando la combinación convexa sólo es posible obtener los puntos soportados A y B, incluso si se intenta realizar la parametrización completa sobre el vector de pesos λ.

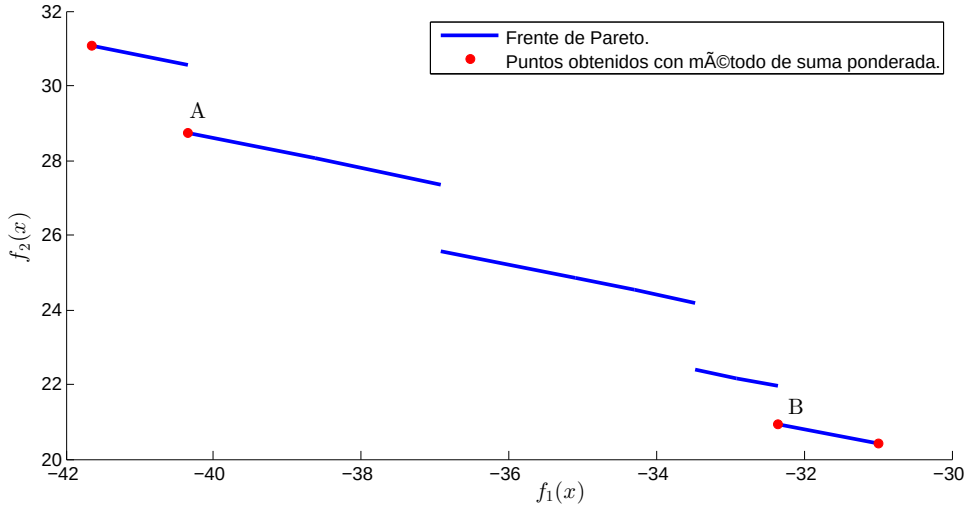

Figura 2.6: Frente de Pareto correspondiente al Ejemplo 2.2.1, donde además se muestran los puntos obtenido a través de la suma de pesos ponderada.

Los primeros intentos por dar solución a los MOMIPs usando el método de suma ponderada no fueron muy exitosos pues únicamente se obtienen puntos de Pareto soportados y no todas las regiones del frente, lo que es una clara desventaja si el objetivo es tener una amplia gama de puntos del frente de Pareto de los cuales el tomador de decisiones puede seleccionar el ó los que más se adecuen a sus requerimientos y necesidades. Incluso si se esta satisfecho con obtener solo los puntos soportados, el problema mixto que se debe resolver se considera un problema que pertenece a la clase $\mathbb{NP}$ [Garey and Johnson, 1990]. Cabe mencionar que este hecho no solo es una desventaja del método de suma ponderada sino de cualquier método de naturaleza puntual pues para cada configuración distinta de los parámetros se requiere resolver un problema mixto.

2.2.1. Estado del arte

En la actualidad existen pocos estudios que presenten un algoritmo para encontrar el frente de Pareto de un MOMIP. La mayoría se basan en la siguiente observación. Si consideramos el conjunto S como la proyección del conjunto factible Ω sobre el espacio de variables enteras, entonces podemos fijar las variables enteras a algún valor $s \in S$ y transformar el problema MOMIP a un MOLP y así aplicar con facilidad el método Multisimplex que se describe en la sección 2.1.2. Luego denotamos por $\mathcal{P}_f(s)$

al frente de Pareto resultante del MOLP donde las variables enteras se encuentran fijas al valor s. Es claro, entonces que el P_f del problema multiobjetivo mixto original se puede obtener al determinar los puntos no dominados del conjunto $\bigcup_{s\in S}\mathcal{P}_f(s)$. El siguiente ejemplo nos ayudará a ilustrar la idea de obtener el frente de Pareto a través del conjunto $\bigcup_{s\in S}\mathcal{P}_f(s)$.

Ejemplo 2.2.2.

$$\begin{aligned}
\text{mín}\quad & F(x) = \begin{pmatrix} -8x_1 - 7x_2 + x_3 + x_4 - 8x_5 - 8x_6 - 9x_7 - 3x_8 \\ -2x_1 - 9x_2 - 4x_3 - 9x_4 - 4x_5 - 3x_6 - 3x_7 + x_8 \end{pmatrix} \\
\text{s. a}\quad & \begin{aligned} 15x_1 + 10x_2 + 9x_3 + 19x_4 + 19x_5 + 3x_6 + 10x_7 + 11x_8 &\leq 200 \\ 18x_1 - 5x_2 - 7x_3 + 19x_4 + 5x_5 + 18x_6 - 9x_7 + 11x_8 &\leq 120 \\ -7x_1 + 11x_2 - 2x_3 - 6x_4 + 14x_5 + 14x_6 + 16x_7 + 13x_8 &\leq 100 \\ 18x_1 - 10x_2 + 6x_3 + 20x_4 - 6x_5 + 19x_6 + 18x_7 + 2x_8 &\leq 134 \\ x_1,\ x_2 \in \{0,1\} & \end{aligned}
\end{aligned} \tag{2.21}$$

Como se muestra en la descripción del problema, las variables x_1 y x_2 son enteras y binarias, entonces se pueden considerar cuatro combinaciones distintas de los valores que cada variable entera puede tomar. Cada combinación la denotamos como

$$\begin{aligned} s_1 &= \{0,0\} \\ s_2 &= \{1,0\} \\ s_3 &= \{0,1\} \\ s_4 &= \{1,1\} \end{aligned}$$

Así el conjunto de proyección queda definido por $S = \{s_1, s_2, s_3, s_4\}$. Para cada elemento de S se aplica el método Multisimplex pues las variables enteras ya tienen un valor fijo, de manera que el problema únicamente tiene variables continuas.

En la Figura 2.7 se observa el frente de Pareto obtenido para cada elemento de S. Finalmente consideremos la unión de estos conjuntos, es decir, $\bigcup_{i=1}^{4}\mathcal{P}_f(s_i)$ y determinemos los puntos que son no dominados. Con este procedimiento se obtiene el frente de Pareto que se muestra en la Figura 2.8 y que correspondiente a la solución del Ejemplo 2.2.2. Vale la pena señalar que las combinaciones s_3 y s_4 son las únicas que contribuyen a la generación de este frente.

A la fecha la mayoría de los algoritmos desarrollados para MOMIP se han restringido al caso binario. Paternak y Passy fueron los primeros en presentar un artículo sobre optimización mixta usando variables binarias y examinando dos funciones

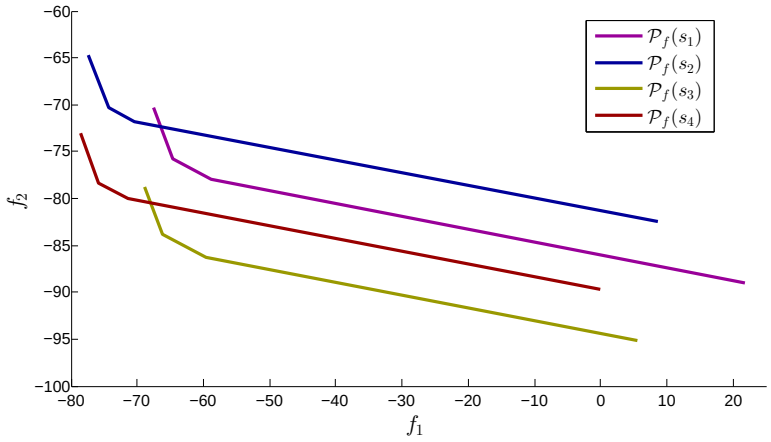

Figura 2.7: Frente de Pareto para cada combinación de valores en las variables enteras del Ejemplo 2.2.2. El frente de Pareto global del problema se forma con las componentes $\mathcal{P}_f(s_3)$ y $\mathcal{P}_f(s_4)$.

objetivo [Pasternak and Passy, 1973]. Para solucionar el problema desarrollan un algoritmo basado en un enfoque paramétrico combinado con una variante del método de filtro de Balas[1] [Egon Balas, 1965].

Después, hasta 1998, Mavrotas y Diakoulaki proponen un algoritmo cuyo objetivo es generar todo el conjunto de Pareto de un problema multiobjetivo mixto con variables binarias [Mavrotas and Diakoulaki, 1998]. Sin embargo, los autores argumentan que cuando el tamaño del problema es muy grande[2] la generación de todo el conjunto de Pareto es prácticamente imposible y se sugiere la incorporación de dos técnicas que ayudan al tomador de decisiones a enfocar la búsqueda de acuerdo a sus preferencias y obtener una adecuada aproximación del conjunto de Pareto. La primera técnica consiste en introducir cotas sobre las funciones objetivo con el fin de descartar soluciones que no son interesantes, la segunda se basa en un proceso de filtro que permite identificar un subconjunto representativo del conjunto de Pareto. Si se sigue la recomendación de los autores el algoritmo se clasifica como uno interactivo, pues en cierta etapa se debe consultar al tomador de decisiones para dirigir la búsqueda.

[1]Método usado en optimización monobjetivo para resolver problemas mixtos con variables binarias, el cual era el más usado antes del desarrollo del método de ramificación y acotamiento.

[2]Los autores consideran un problema como grande cuando éste tiene más de 50 variables y/o 50 restricciones.

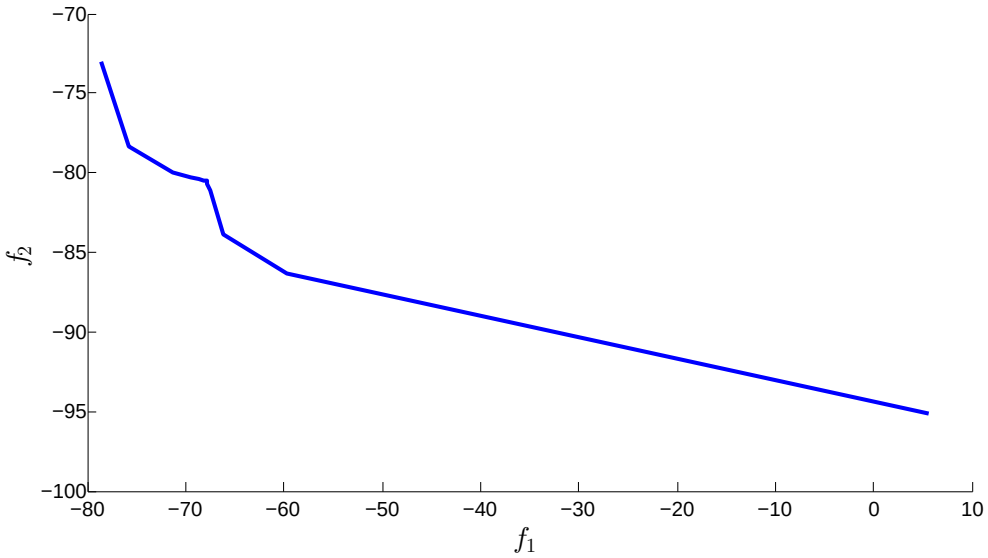

Figura 2.8: Conjunto obtenido al seleccionar los puntos no dominados del conjunto $\bigcup_{i=1}^{4} \mathcal{P}_f(s_i)$ y que corresponde al frente de Pareto del Ejemplo 2.2.2.

A pesar de que el *método de ramificación y acotamiento* (BBM)[3] esta diseñado para problemas monobjetivo Mavrotas y Diakoulaki son los primeros en incorporarlo a un algoritmo para problemas multiobjetivo. El método examina, con la ayuda de un árbol, todas las posibles combinaciones de las variables discretas, descartando aquellas que no son factibles u óptimas. Sin embargo, el método convencional esta diseñado para encontrar el óptimo de un problema monobjetivo y no es capaz de considerar varios objetivos. La noción de optimalidad que se usa en monobjetivo se cambia por la de dominancia para problemas multiobjetivo. Así el método convencional de ramificación y acotamiento se modifica para encontrar todos los puntos no dominados del problema multiobjetivo mixto después de examinar todas las posibles combinaciones de las variables binarias.

Como en el caso monobjetivo el árbol de búsqueda se construye durante el proceso de solución con tal de representar todas las posibles combinaciones de los valores en las variables binarias. Cada nodo del árbol representa un subproblema con algu-

[3]Por sus siglas en inglés Branch and Bound Method.

nas variables binarias fijas a un valor. Las variables binarias sin un valor asignado se denominan *libres*. Como se muestra en la Figura 2.9 el nodo raíz corresponde al problema completamente relajado, donde todas las variables discretas son libres.

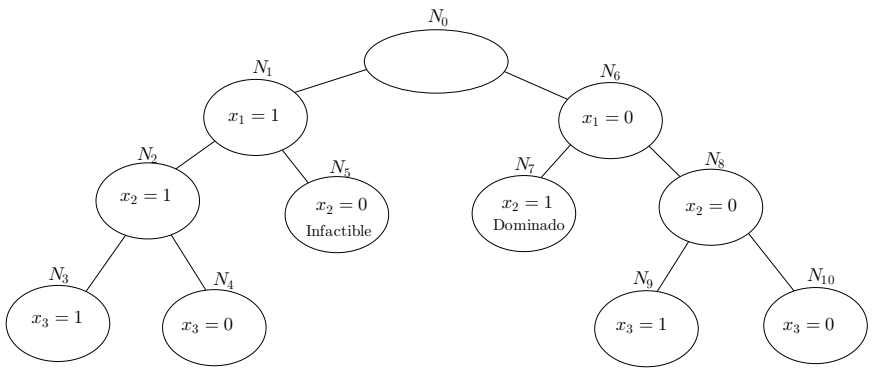

Figura 2.9: Frente de Pareto para cada combinación de valores en las variables enteras del Ejemplo 2.2.2.

La modificación principal al BBM esta relacionada con el diferente concepto de optimalidad entre problemas monobjetivo y multiobjetivo. Mientras en problemas monobjetivo la solución es única, en multiobjetivo se tiene un conjunto de vectores compromiso o no dominados. Así en cada nodo del BBM convencional la mejor solución encontrada hasta el momento o simplemente *solución titular*, se almacena, actualiza, se usa para podar ramas del árbol que conducen a valores peores o infactibles y es finalmente el valor de salida cuando todos los nodos activos se han examinado. En cambio, en la versión multiobjetivo de Mavrotas y Diakoulaki el concepto de solución titular se cambia por *lista titular* debido al hecho de que hay más de un vector no dominado. El propósito de esta lista es similar pues se actualiza constantemente al atravesar el árbol y mientras se usa para filtrar soluciones dominadas ésta consta únicamente de vectores no dominados. Al final del algoritmo la lista titular debe contener las combinaciones que potencialmente forman parte del frente de Pareto. Entonces para cada combinación se resuelve un MOLP, pues las variables binarias ya tienen un valor asignado. Al final se realiza un filtro para mantener solo soluciones no dominadas por lo que le resultado final debe ser igual al frente de Pareto o al menos contener un subconjunto que permita construir el frente completo.

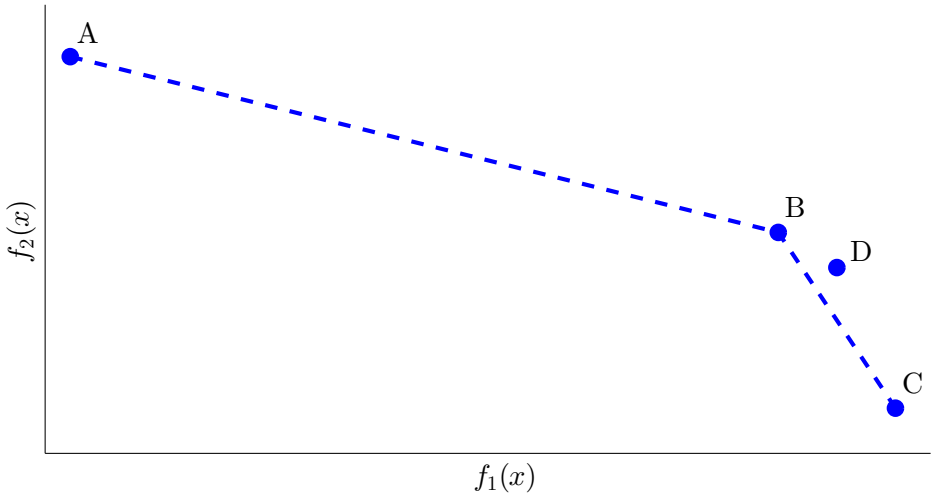

Figura 2.10: Los puntos B y C no dominan por sí solos al punto D, sin embargo si es dominado por una combinación convexa de ellos.

En el artículo [Mavrotas and Diakoulaki, 2005b], los autores introducen varias componentes que mejoran el algoritmo descrito en [Mavrotas and Diakoulaki, 1998]. En primer lugar se separa el proceso de búsqueda en dos fases con el fin de permitir al tomador de decisiones acelerar la búsqueda hacia las soluciones con mayor preferencia. Además los autores muestran que su esquema de actualización para la lista titular esta mal formulado, pues sólo toma en consideración los puntos no dominados que son extremos. Por ejemplo en la Figura 2.10 los puntos A, B y C son puntos no dominados y extremos. El punto D es no dominado respecto a los puntos B y C. Sin embargo la combinación convexa de éstos sí domina al punto D. Esta observación muestra que la solución final obtenida mediante este proceso pudiera contener puntos que no forman parte del frente de Pareto. Para resolver el problema, los autores proponen un método de filtro que detecte y elimine vectores dominados de la lista final de soluciones. Sin embargo, a pesar de la introducción de estas componentes su algoritmo mejorado sigue teniendo deficiencias.

En [Vincent et al., 2013] se retoma el algoritmo de ramificación y acotamiento para MOMIPs con variables binarias de Mavrotas y Diakoulaki. En este trabajo se observa que aun con las correcciones que hacen Mavrotas y Diakoulaki a su algoritmo éste deja fuera de la lista de soluciones regiones que pueden ser parte del frente de

Pareto. Esto porque no consideran todos los casos en que alguna combinación convexa puede dominar un punto o parte de otra combinación convexa. Además el filtro final que se realiza en el algoritmo es un proceso separado de la enumeración en el árbol. De hecho las combinaciones eficientes se deducen después de haber examinado todos los nodos del árbol. Algunos puntos extremos pueden ser removidos del proceso y sin embargo cierta combinación convexa puede formar parte del frente de Pareto. Por lo que al final se obtiene un conjunto de soluciones incompleto.

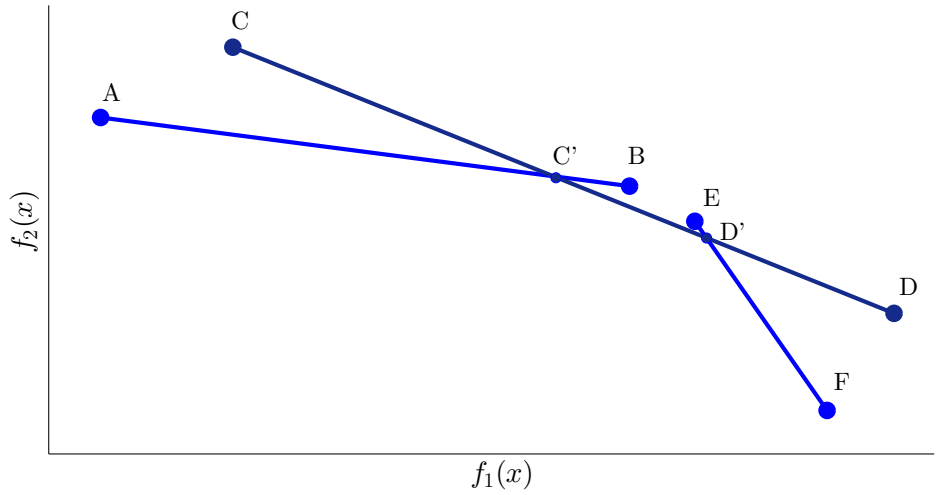

Figura 2.11: La linea entre C' y D' queda fuera del resultado final debido a que C y D son descartados con anterioridad por el algoritmo de Mavrotas y Diakoulaki.

Para mostrar el problema, consideremos el ejemplo de la Figura 2.11. De acuerdo al algoritmo Mavrotas y Diakoulaki, los puntos extremos B y E se mantienen en la lista titular aunque son dominados por cierta combinación convexa de los puntos C y D. El método de filtro de Mavrotas falla, en este caso, porque C y D se han descartado del análisis antes debido a que son dominados por A y F, respectivamente. Por lo tanto no están almacenados en la lista titular y no es posible considerar su información en el proceso de filtro final. Podemos notar además que parte de la combinación convexa entre C y D que se pierde del análisis es no dominada. Así pues en [Vincent et al., 2013] se propone considerar todos los puntos extremos que surjan usando una representación del conjunto de puntos no dominados asociado a un proceso de actualización apropiado para la lista titular.

Recientemente, en [Boland et al., 2014] se presentó un método que encuentra el frente y el conjunto de Pareto exactos de un BOMIP. El método se denomina *Triangle Splitting Method.*

2.2.2. Triangle splitting method

El *triangle splitting method* (TSM) es un método, propuesto recientemente en [Boland et al., 2014], que pretende obtener el frente de Pareto de un BOMIP. El método resuelve ciertos problemas de optimización escalar usando como espacio de búsqueda el espacio de los objetivos. Por ello, una de sus componentes esenciales es el uso de algún software para optimización lineal mixta como `FICO® Xpress`[fic, 2016] [Daniel, 2009], `Gurobi® Optimizer`[gur, 2015] o `lp_solve®`[Berkelaar et al., 2004].

La idea central es buscar puntos no dominados explotando las propiedades lineales del problema y restringiendo el espacio objetivo a partir de dos puntos no dominados dados.

Definición 2.6. Sean $F^1 = (f_1^1, f_2^1)^T, F^2 = (f_1^2, f_2^2)^T \in \mathcal{P}_f$, dos puntos no dominados tales que $f_1^1 \leq f_1^2$, $f_2^2 \leq f_2^1$ de modo que $F^1 \neq F^2$. Se define como $R(F^1, F^2)$[4] al rectángulo formado por F^1 y F^2. Asimismo, $T(F^1, F^2)$[5] define el triangulo superior derecho de $R(F^1, F^2)$ y $H(F^1, F^2)$ es la hipotenusa de tal triángulo, es decir, el segmento de recta que une los puntos F^1 y F^2.

El funcionamiento del TSM se basa en restringir el espacio de búsqueda dentro del rectángulo o triángulo definido a partir de dos puntos no dominados, F^1 y F^2. El objetivo es obtener otros puntos no dominados que aún no se conocen ó determinar si todos los puntos sobre $H(F^1, F^2)$ son no dominados. Esto da lugar a establecer dos tipos de búsqueda: en un rectángulo o en un triángulo. Cada tipo tiene distintos propósitos y realizan diferentes operaciones las cuales se describirán a lo largo de esta sección. Sin embargo, es importante mencionar que el método debe mantener una lista de prioridad, donde cada elemento de ésta el tipo de búsqueda empleada y el tipo de división: horizontal o vertical. El Algoritmo 2.2.1 presenta una descripción, un pseudocódigo, del método TSM, donde la lista de prioridad se inicializa con una búsqueda en el rectángulo definido por los puntos óptimos de cada función objetivo

[4]Formalmente los cuatro vértices del rectángulo $R(F^1, F^2)$ son $(f_1^1, f_2^1), (f_1^1, f_2^2), (f_1^2, f_2^1)$ y (f_1^2, f_2^2). Sin embargo, para la notación sólo se usan los puntos que son parte del $\mathcal{P}_f$.

[5]Los vértices del triángulo son $(f_1^1, f_2^1), (f_1^2, f_2^1)$ y (f_1^2, f_2^2), pero al igual que en el rectángulo sólo se usan los puntos que son parte de $\mathcal{P}_f$.

y con tipo de división horizontal.

Algoritmo 2.2.1 Descripción del algoritmo para el método TSM

Salida: Conjunto de puntos no dominados pertenecientes a $\mathcal{P}_f$.

1: Resolver el problema $\min\limits_{x\in\Omega} f_1(x)$
   Sea x^1 el vector que minimiza $f_1(x)$.
2: Resolver el problema $\min\limits_{x\in\Omega} f_2(x)$
   Sea x^2 el vector que minimiza $f_2(x)$.
3: $F^1 \leftarrow Cx^1$
4: $F^2 \leftarrow Cx^2$
5: $listaprioridad = (R(F^1, F^2), horizontal))$. ▷ Inicializar lista de prioridad
6: **while** $listaprioiridad \neq \emptyset$ **do**
7: Actualizar la lista de prioridad mediante la exploración de acuerdo al tipo de búsqueda (rectángulo o triángulo).
8: **end while**

La exploración de un rectángulo consiste en buscar nuevos puntos no dominados. Supongamos que conocemos los puntos F^1 y F^2 y por lo tanto podemos construir la recta que pasa sobre tales puntos y cuya pendiente es $m_z = \frac{f_2^1 - f_2^2}{f_1^1 - f_1^2}$. Por otra parte, esta misma recta se puede ver como una curva de nivel definida como en la ecuación (2.22), donde por ejemplo el valor de la curva de nivel que pasa exactamente sobre F^1 y F^2 es $z_h = (f_2^1 - f_2^2)f_1^1 + (f_1^2 - f_1^1)f_2^1$.

$$z(x) = (f_2^1 - f_2^2)f_1(x) + (f_1^2 - f_1^1)f_2(x) \tag{2.22}$$

En la Figura 2.12 se pueden apreciar los puntos $F^1, F^2 \in \mathcal{P}_f$, la recta que los une z_h y distintas curvas de nivel $z(x)$. Con ayuda de la imagen se puede observar que si existe x^* tal que minimiza a $z(x)$ dentro de la región factible y el rectángulo $R(F^1, F^2)$ y además se cumple la siguiente desigualdad

$$z(x^*) < (f_2^1 - f_2^2)f_1^1 + (f_1^2 - f_1^1)f_2^1 \tag{2.23}$$

entonces x^* es un punto óptimo de Pareto y por lo tanto $F^* = F(x^*)$ forma parte del frente de Pareto.

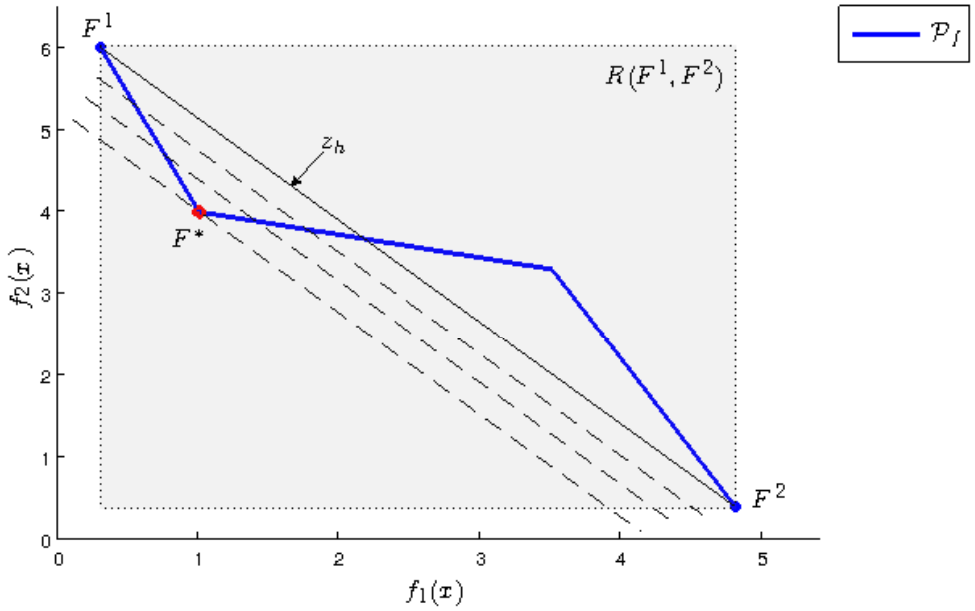

Figura 2.12: Búsqueda de puntos no dominados en $R(F^1, F^2)$

Así pues, la búsqueda de tipo rectángulo inicia resolviendo el subproblema

$$\begin{array}{ll} \min\limits_{x\in\Omega} & z(x) = (f_2^1 - f_2^2)f_1(x) + (f_1^2 - f_1^1)f_2(x) \\ s.\ a & F(x) \in R(F^1, F^2) \end{array} \tag{2.24}$$

cuya solución o bien corresponde a un nuevo punto no dominado o es una combinación convexa de F^1 y F^2. Para distinguir el caso, se verifica si la desigualdad (2.23) se cumple. Si es así, entonces $x^* \in \mathcal{P}_s$, $F^* \in \mathcal{P}_f$ y se aplica el método de forma recursiva para buscar ahora sobre dos nuevos rectángulos: $R(F^1, F^*)$ y $R(F^*, F^2)$.

Cuando la desigualdad (2.23) no se cumple significa que no existen elementos de $\mathcal{P}_f$ por debajo de la linea entre F^1 y F^2. Formalmente, el conjunto $\{(F(x)-\mathbb{R}^2_>)\cap\mathcal{P}_f : F(x) \in H(F^1, F^2)\}$, donde $\mathbb{R}^2_> = \{x \in \mathbb{R}^2 : x > 0\}$ representa el cuadrante positivo de $\mathbb{R}^2$, es un conjunto vacío. Entonces la búsqueda continua ahora en el triángulo $T(F^1, F^2)$. El Algoritmo 2.2.2 muestra una descripción con pseudocódigo de como se realiza la exploración de un rectángulo. Retomando la Figura 2.12 notemos que aunque la recta que une F^1 con F^* pertenece al frente de Pareto, la búsqueda sobre el rectángulo sólo devuelve el punto F^*.

Algoritmo 2.2.2 Descripción del algoritmo para la búsqueda de rectángulo

Entrada: $(R(F^1, F^2), direccion)$
$F(x)$, función objetivo.
Salida: Nueva lista de prioridad con triángulos por explorar.

1: Resolver $\min_{x\in\Omega} \quad z(x) = (f_2^1 - f_2^2)f_1(x) + (f_1^2 - f_1^1)f_2(x)$
Sea x^* el vector que minimiza $z(x)$.
2: **if** $z(x^*) < (f_2^1 - f_2^2)f_1^1 + (f_1^2 - f_1^1)f_2^1$ **then**
3: Explorar recursivamente los siguientes rectángulos:
$(R(F^1, F^*), direccion)$ y $(R(F^*, F^2), direccion)$
4: **else**
5: Agregar a la lista de prioridad el siguiente elemento:
$(T(F^1, F^2), direccion)$
6: **end if**

Teorema 2.6. *[Boland et al., 2014] Sean F^1, F^2 dos puntos no dominados en el espacio objetivo. Si el conjunto $\{(F(x)-\mathbb{R}^2_>)\cap\mathcal{P}_f : F(x) \in H(F^1, F^2)\} = \emptyset$ y existen $x \in \mathbb{Z}^{n_e}$ y $x^1, x^2 \in \mathbb{R}^{n_c}$ tal que $(x, x^1), (x, x^2) \in \Omega$, $F((x, x^1)) \leq F^1$ y $F((x, x^2)) \leq F^2$, entonces $H(F^1, F^2)\} \subset \mathcal{P}_f$.*

El Teorema 2.6 implica que para el triángulo $T(F^1, F^2)$ tal que $\{(F(x)-\mathbb{R}^2_>)\cap\mathcal{P}_f : F(x) \in H(F^1, F^2)\} = \emptyset$ se puede determinar si $H(F^1, F^2) \subset \mathcal{P}_f$. Esto con apoyo del subproblema

$$\begin{aligned} \text{máx} \quad & f_1((x, x^2)) \\ s.\ a \quad & F((x, x^1)) \leq F^1 \\ & \alpha_1 f_1((x, x^2)) + \alpha_2 f_2((x, x^2)) = \alpha_1 f_1^1 + \alpha_2 f_2^1 \\ & (x, x^1), (x, x^2) \in \Omega. \end{aligned} \tag{2.25}$$

donde $\alpha_1 = f_2^1 - f_2^2$ y $\alpha_2 = f_1^2 - f_1^1$.

Cabe destacar que la segunda restricción en el problema (2.25) garantiza que el punto se encuentre sobre la recta imaginaria que une a F^1 con F^2. Además, notemos que si el valor óptimo es mayor o igual que f_1^2, entonces F^1 y F^2 cumplen las condiciones del Teorema 2.6 y por lo tanto se puede asegurar que todos los puntos sobre la hipotenusa del triángulo $T(F^1, F^2)$ son puntos no dominados, es decir $H(F^1, F^2) \subset \mathcal{P}_f$.

Cuando $H(F^1, F^2)$ no es parte de $\mathcal{P}_f$ el triángulo se dividirá respecto al tipo de dirección definido en la lista de prioridad, ya sea mediante la división del triángulo horizontalmente en $\frac{f_2^1+f_2^2}{2}$ o dividiendo el triángulo verticalmente en $\frac{f_1^1+f_1^2}{2}$. Más específicamente, cuando el triángulo se divide horizontalmente (si se divide verticalmente el procedimiento es análogo), se inicia por calcular $\bar{F}^1$ a través de la solución del siguiente problema[6]

$$\bar{F}^1 = \operatorname*{lex\,mín}_{x\in\Omega} \left\{ f_1(x), f_2(x) : f_2(x) \leq \frac{f_2^1 + f_2^2}{2},\ F(x) \in T(F^1, F^2) \right\} \tag{2.26}$$

Si ocurre que $\bar{F}^1$ se encuentra sobre la línea de corte, es decir, $\bar{F}_2^1 = \frac{f_2^1+f_2^2}{2}$, es claro ver que $\bar{F}^2 = \bar{F}^1$, por lo que no es necesario calcular $\bar{F}^2$. En caso contrario calculamos $\bar{F}^2$ como

$$\bar{F}^2 = \operatorname*{lex\,mín}_{x\in\Omega} \left\{ f_2(x), f_1(x) : f_1(x) \leq \bar{F}_1^1,\quad F(x) \in T(F^1, F^2) \right\} \tag{2.27}$$

Una vez que se encuentran los puntos $\bar{F}^1$ y $\bar{F}^2$ se definen dos nuevos rectángulos $R(F^1, \bar{F}^2)$ y $R(\bar{F}^1, F^2)$ que se agregan a la lista de prioridad. Sin embargo es importante notar que cuando $\bar{F}^2 = F^1$, el rectángulo $R(F^1, \bar{F}^2)$ consta únicamente del punto F^1 y no es necesario explorarlo (lo mismo ocurre si $\bar{F}^1 = F^2$). Asimismo, cuando se agregan rectángulos a la lista de prioridad, se invierte el tipo de dirección con la cual fueron generados. Por ejemplo, si se obtuvieron de dividir un triángulo horizontalmente, entonces los nuevos rectángulos tendrán tipo de división vertical.

En el Algoritmo 2.2.3 se muestra una descripción con pseudocódigo de como se realiza la exploración de un triángulo. Primero resolviendo el subproblema definido por (2.25). Se verifica si la solución obtenida es mayor o igual a f_1^2, de acuerdo al Teorema 2.6. En caso afirmativo se puede concluir que los puntos sobre la hipotenusa del triángulo bajo análisis pertenecen al frente de Pareto. En caso contrario se realiza la división del triángulo y se resuelven los problemas (2.26) y (2.27) con el fin de obtener los puntos $\bar{F}^1$ y $\bar{F}^2$, respectivamente. Al final se definen dos nuevos rectángulos, terminando el proceso sobre el triángulo.

Para complementar el procedimiento en la Figura 2.13 se ejemplifica la búsqueda del triángulo. En la primer imagen se muestra que $H(F^1, F^2)$ no es parte del frente

[6]Recordar que en la sección 1.1.1 se define el método lexicográfico donde se usa la notación lex mín.

Algoritmo 2.2.3 Descripción del algoritmo para la búsqueda de triángulo

Entrada: $(T(F^1, F^2), direccion)$
$F(x)$, función objetivo.
Salida: Lista con puntos no dominados.

1: Resolver el subproblema (2.25). Sea f_1^* el valor óptimo.
2: **if** $f_1^* \geq f_1^2$ **then**
3: Actualizar lista de puntos no dominados, $H(F^1, F^2) \in \mathcal{P}_f$.
4: **else**
5: Dividir $T(F^1, F^2)$ de acuerdo a la *direccion*.
6: Calcular $\bar{F}^1$ y $\bar{F}^2$.
7: Invertir *direccion* y agregar a la lista de prioridad los rectángulos correspondientes: $(R(F^1, \bar{F}^2), direccion)$ y $(R(\bar{F}^1, F^2), direccion)$
8: **end if**

por lo que se sigue la búsqueda hasta encontrar los puntos $\bar{F}^1$ y $\bar{F}^2$, los cuales se muestran en la segunda imagen. A partir de ese momento se definen nuevos rectángulos que se agregan a la lista de prioridad.

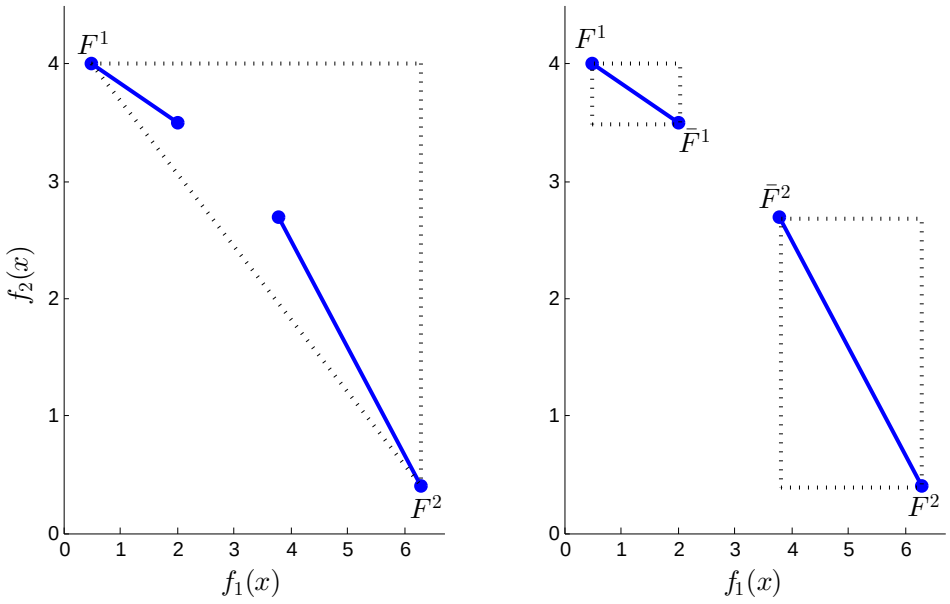

Figura 2.13: Búsqueda de puntos no dominados en $T(F^1, F^2)$

Después de analizar y llevar a la practica este método, experimentando sobre determinados problemas de prueba, pudimos determinar que posee las siguientes ventajas:

1. Es un método fácil de entender e implementar, pues la idea principal es clara. Se trata de un método deterministico.
2. En cada etapa del método los puntos que se obtienen son parte del frente del Pareto.
3. Transforma el problema biobjetivo en varios problemas de optimización escalar.
4. En principio debería ser capaz de encontrar completamente el frente de Pareto, sin embargo en la practica el resultado depende de las condiciones del problema y del paquete de optimización mixta que se elija.

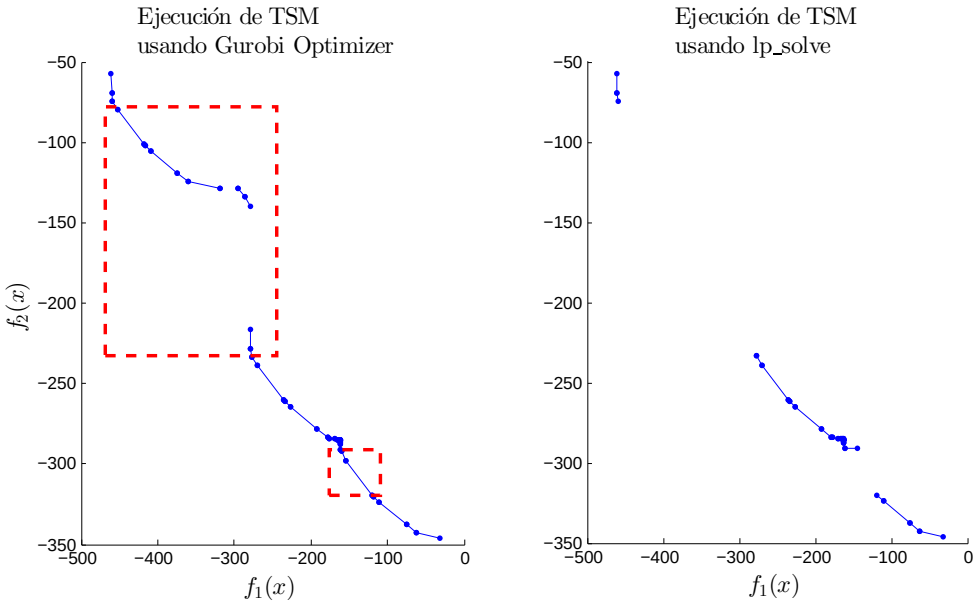

Figura 2.14: Resultados del TSM usando `Gurobi® Optimizer` y `lp_solve®` para el mismo problema.

Sin embargo las desventajas que se identificaron son:

1. Para mantener la factibilidad de las soluciones enteras y resolver los subproblemas auxiliares de optimización mixta se requiere utilizar software especializado. Esto puede resultar en una desventaja pues el desempeño del método

dependerá del software elegido. Por ejemplo, en la Figura 2.14 se muestran dos diferentes resultados del método TSM para el mismo problema. En la imagen de la izquierda se utilizó `Gurobi® Optimizer` mientras que para la imagen de la derecha a `lp_solve®`. Como se observa, cuando se usa `Gurobi® Optimizer` se obtiene todo el frente de Pareto a diferencia de cuando se usa `lp_solve®`. El recuadro encierra las soluciones que `lp_solve®` no es capaz de obtener pero `Gurobi® Optimizer` sí.

2. Cada subproblema que se resuelve resulta ser un problema de optimización mixta, que como se ha reportado en la literatura pertenece a la clase NP-Completo. Entonces puede resultar que el método devuelva una solución subóptima, que durante la búsqueda se pierdan regiones del frente de Pareto o que se consuman demasiados recursos computacionales y el tiempo de ejecución sea elevado.

Así pues, el método TSM es una heurística determinista que en la práctica presenta mayor potencial que los métodos previamente estudiados. En el siguiente capítulo presentaremos nuestra propuesta de heurística poblacional para abordar MOMIPs basada en computación evolutiva, y mostraremos de manera empírica que la hibridación del método TSM con un algoritmo genético produce resultados competitivos y eficientes para una *suite* determinada de problemas de prueba.

Capítulo 3

Algoritmo Genético para MOMIP

3.1. Algoritmos Genéticos

Los algoritmos que constituyen la Computación Evolutiva (llamados algoritmos genéticos) son heurísticas inspiradas en los mecanismos que guían la evolución biológica. Estas técnicas son utilizadas para generar aproximaciones en problemas de optimizacón y búsqueda. Dado un problema especifico, la entrada del algoritmo es un conjunto de posibles soluciones. Las soluciones se evalúan considerando valores dados por una función de adaptación para seleccionar a las más prometedoras. A partir de éstas el algoritmo genera nuevas soluciones, comenzando así un proceso iterativo que finaliza hasta alcanzar un cierto criterio predefinido[Eiben and Smith, 2010].

Los AGs se llama así por su inspiración en la teoría Neodarwinista[Rothlauf, 2002]. Según esta teoría los fenómenos evolutivos se explican por medio de la integración de la teoría de la evolución por selección natural de Charles Darwin[Darwin, 1859] y la teoría genética de Gregor Mendel[Mendel, 1866][Bateson, 1901]. Es decir, la evolución se debe a la acumulación de pequeñas mutaciones favorables preservadas por la selección natural. Por consiguiente, la producción de nuevas especies no sería más que la extrapolación y magnificación de las variaciones que ocurren dentro de la especies. En conclusión la filosofía de dicho algoritmo se basa en los siguientes principios:

- Existe una población de individuos con diferentes propiedades y habilidades. Además hay un límite superior para el número de individuos en una población.
- Herencia genética. La naturaleza crea individuos con características similares a los individuos existentes.

- Selección del más apto. Los individuos más prometedores se seleccionan con mayor frecuencia para reproducción por selección natural.
- Mutación genética aleatoria. La mutación genética se produce de manera aleatoria. Si las mutaciones mejoran las características del individuo éstas se preservan.

Entre 1960 y 1970 varios científicos estudiaron sistemas evolutivos con la idea de que la evolución podría usarse como una herramienta de optimización. La idea era hacer evolucionar soluciones usando operadores de variación genética y selección natural. Durante el periodo se implementaron 3 diferentes variantes de esta idea principal. En Estados Unidos, Lawrence Fogel, Owens y Walsh introdujeron el concepto de Programación Evolutiva [Fogel et al., 1966] al utilizar la evolución simulada como un proceso de aprendizaje. Su objetivo era generar inteligencia artificial. Mientras tanto Ingo Rechenberg y Hans-Paul Schwefel crearon las Estrategias Evolutivas [Rechenberg, 1973, Schwefel, 1995], técnicas para aproximarse a la solución de un problema de optimización. Finalmente los *algoritmo genético*s (GAs) fueron inventados por John H. Holland [Holland, 1975]. Holland, sus estudiantes y algunos colegas estudiaron y desarrollaron los GA en la Universidad de Michigan. El objetivo principal de Holland no era diseñar un algoritmo para resolver un problema en particular. Deseaba estudiar formalmente el fenómeno de selección y adaptación de la naturaleza con el próposito de encontrar maneras en las que los mecanismos de adaptación natural pudieran ser importados a un sistema de computación. En 1975 presentan el primer *algoritmo genético* como una abstracción de la evolución biológica y dan un marco teórico de la adaptación dentro del GA [Holland, 1975].

El algoritmo de Holland itera de una población a otra a partir de un mecanismo de "selección natural" junto con operadores inspirados en la genética. El operador de selección elige aquellos elementos en la población que serán capaces de reproducirse. En promedio los que se han adaptado mejor se reproducen con más frecuencia que el resto de los elementos. Por más de 15 años la Programación Evolutiva, las Estrategias Evolutivas y los Algoritmos Genéticos se estudiaron y desarrollaron por separado. En 1990 surgió una cuarta corriente que seguia las mismas ideas de evolución, la Programación Genética creada por Koza [Koza, 1991, Koza, 1994]. A partir de entonces las cuatro vertientes se han visto como parte del campo de estudio llamado Computación Evolutiva [Eiben and Smith, 2010].

En la Figura 3.1 se muestra el diagrama de flujo general que sigue un GA. Los GA inician generando un conjunto de posibles soluciones que se conoce como *Po-*

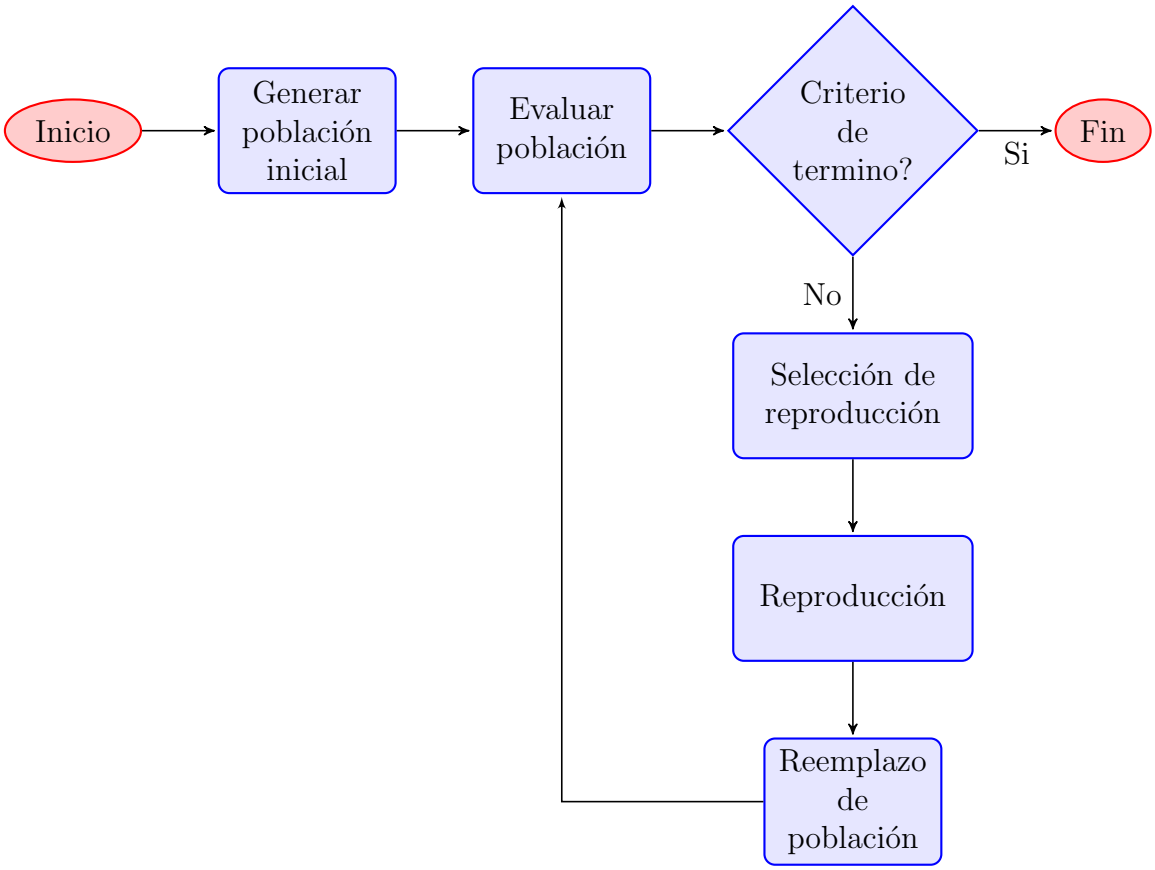

Figura 3.1: Flujo General de los Algoritmos Genéticos

blación. Con frecuencia la población inicial se genera aleatoriamente, aunque si se hace de cualquier otra manera se debe garantizar que dentro de la población inicial se tenga diversidad de soluciones. Cada solución representa un *Individuo* dentro de la población manejada por el GA. Cada individuo se evalúa mediante una función de adaptación definida previamente para determinar que tan buena es la solución. Con este paso se mide la adaptación de un individuo al ambiente (problema). Basado en su adaptación se eligen individuos como semilla para generar nuevos individuos. Con ello se espera que los genes de los buenos individuos pasarán hacia futuras generaciones. A continuación se determina qué individuos deben continuar dentro de la población. Se continua evaluando y generando nuevos individuos mediante la reproducción hasta alcanzar un criterio que determina el fin del algoritmo.

Definición 3.1. Se denomina *cromosoma* a la estructura de datos que contiene una cadena de parámetros del problema a resolverse.

Definición 3.2. En un AG se conoce como *gen* a la sección de un cromosoma que codifica el valor de un solo parámetro.

Definición 3.3. Se denomina individuo a un solo miembro de la población de soluciones potenciales a un problema. Cada individuo contiene un cromosoma que representa una solución posible al problema a resolverse.

Definición 3.4. Se denomina Población a un conjunto finito de individuos.

3.2. Algoritmo genético para MOMIP

La elección de usar un AG entre otras heurísticas se debe a su carácter poblacional pero sobre todo al uso y manejo de buenos *bloques constructores*[Holland, 1975]. En la Sección 2.2.1 se muestra que para resolver un MOMIP basta con obtener las combinaciones de valores sobre las variables discretas que forman el frente de Pareto. A diferencia de los métodos estado del arte, que en cada iteración analizan sólo una combinación, nuestro algoritmo, denominado MOMIPGA, aprovecha su característica de algoritmo poblacional y analiza en cada iteración varias combinaciones. Además, el uso de buenos bloques constructores garantiza que buenas combinaciones (es decir aquellas que posiblemente forman el frente de Pareto) ayuden a obtener otras combinaciones igual de buenas o mejores. Así en lugar de resolver varios subproblemas ó analizar una combinación por iteración, el algoritmo MOMIPGA usa las propiedades geométricas que se presentan en los MOMIPs y las características de los AG, para así explotar al máximo el espacio de búsqueda y obtener el frente de Pareto. A partir de este momento se considerarán sólo problemas donde hay dos funciones objetivo. Así nuestro algoritmo esta restringido a este tipo de problemas multiobjetivo.

3.2.1. Codificación de las soluciones

La representación es una etapa importante en la definición de un AG. En principio se da la definición de lo que para el MOMIPGA es un individuo y como éste se representa para llevar a cabo el proceso de evolución (búsqueda).

Definición 3.5. Un vector $X \in \mathbb{Z}^{n_e} \times \mathbb{R}^{n_c}$ cuyas entradas corresponden a una posible solución de MOMIP se denomina *individuo*.

En MOMIPGA se hace una distinción entre las variables discretas y las continuas. Las posibles soluciones o individuos se representan a través de sus cromosomas; cadenas de longitud n_c+n_e (recordando que n_c y n_e es el numero de variables continuas y discretas, respectivamente). Cada posición de la cadena almacena el valor de una variable, por lo que la representación es directa. Esto quiere decir que la variable se representa con un valor real si ésta es continua o sino con un valor entero. Además las variables enteras aparecen en las primeras posiciones de la cadena. Esto para facilitar el uso de los operadores de cruza y mutación que, como se expondrá más adelante, se realizan de acuerdo al tipo de variable.

Definición 3.6. Sea X un individuo de MOMIPGA. La cadena que representa a X, $[x_1, \ldots, x_{n_e}, y_1, \ldots, y_{n_c}]$ con $x_i \in \mathbb{Z}$ para $i = 1, \ldots, n_e$ y $y_j \in \mathbb{R}$ para $j = 1, \ldots, n_c$, se denomina *cromosoma*. Asimismo, el valor correspondiente a cada elemento de la cadena se conoce como *gen*.

Uno de los parámetros mas importantes en la definición de un AG es el número de individuos que interaccionan a lo largo de las iteraciones. Este parámetro es conocido como el tamaño de la población, y suele denotarse por N.

3.2.2. Etapa de inicialización de soluciones

Para comenzar la búsqueda se deben asignar valores iniciales a cada individuo. Esto consiste en determinar los genes iniciales de cada uno de los N individuos de la población, es decir, establecer los valores a cada variable de las primeras soluciones. Comúnmente esta inicialización se realiza al azar, asignando números aleatorios a cada variable. Sin embargo, la inicialización puede hacerse a partir de información ya obtenida sobre las variables del problema. En el caso de MOMIPGA la técnica para inicializar los individuos se basa en la idea usada en los problemas de programación mixta. En primera instancia se obtiene una solución considerando todas las variables como continuas, para a partir de ésta construir la solución donde hay n_e variables discretas y n_c variables continuas. Así MOMIPGA hace uso del método bisimplex para inicializar los individuos.

La población inicial se genera a partir de los puntos extremos no dominados que se obtiene del método Simplex para multiobjetivo. La finalidad es generar individuos que se encuentren distribuidos uniformemente a lo largo del frente de Pareto lineal y del hiperplano utópico. Se calcula primero el número de segmentos definidos por estos puntos. A continuación se determina el número de individuos que deben generarse dentro de cada segmento. Así se generar los individuos como combinación

convexa de 2 puntos extremos contiguos, considerando que las combinaciones convexas resultantes queden uniformemente distribuidas. En la Figura 3.2 se muestran los 20 individuos iniciales que se obtienen del frente de Pareto y el hiperplano utópico para un MOMIP. Como se observa, el frente de Pareto contiene cuatro puntos extremos por lo que incluyendo el hiperplano utópico existen 4 segmentos. Entonces se generan en cada segmento 5 individuos para garantizar que la población inicial (de tamaño 20) esta distribuida de forma uniforme en los segmentos, pero además dentro de cada uno de ellos.

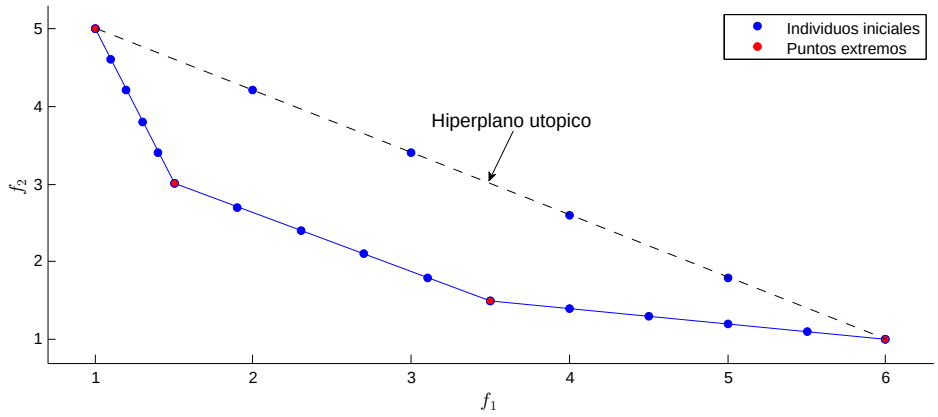

Figura 3.2: Individuos iniciales de acuerdo al frente de Pareto y los puntos extremos obtenidos del método Simplex multi-objetivo.

Hasta ahora, podemos distinguir que los individuos iniciales corresponden a alguna combinación convexa de dos puntos extremos del frente continuo. Las variables discretas aún tienen un valor continuo y por lo tanto los individuos iniciales son infactibles respecto a las restricciones de integridad. Para que estos individuos iniciales sean factibles se usa un mecanismo de reparo que modifica las variables que deben ser discretas pero que por el manejo del AG no lo son. Este mecanismo consiste en hacer un redondeo tomando la parte fraccionaria de la variable como la probabilidad de redondear hacia arriba. Supongamos que se tiene un individuo infactible X cuyo i-ésimo gen x_i no cumple las restricciones de integridad. Para que X sea factible se realiza la siguiente serie de pasos donde $frac(x_i)$ es la parte fraccionaria de x_i

1. Generar un número aleatorio $u \in [0, 1]$.
2. $x_i = \begin{cases} \lfloor x_i \rfloor & \text{si} \quad u \leq frac(x_i). \\ \lfloor x_i \rfloor + 1 & \text{en caso contrario.} \end{cases}$

Por la tanto, el MOMIPGA inicia su ejecución con una población generada con información de las soluciones continuas. Primero se resuelve el problema multiobjetivo prescindiendo de las restricciones de integridad y luego se aplica el mecanismo de reparo sobre aquellas variables que deben ser discretas, garantizando una población inicial factible y lo bastante buena como para comenzar la búsqueda.

3.2.3. Operadores genéticos

Los operadores genéticos que se usan en MOMIPGA son la cruza o recombinación genética y la mutación. La cruza genera nuevos individuos a partir de otros existentes en la población. Sin embargo, no todos los individuos de la población son capaces de realizar tal operación. Ya que el AG está basado en las teorías Neodarwinianas [Howlett, 1989] de la evolución, el MOMIPGA cuenta con un mecanismo de selección que permite a los mejores individuos reproducirse con más éxito que otros. Es así cuando la cruza genera nuevos individuos con los individuos elegidos por el mecanismo de selección. Además, a los individuos generadores se les denomina *padres*, mientras que los individuos que se generan se denominan hijos.

Técnica de cruza

Para cada tipo de variable (continua o discreta) se realiza un tipo de cruza distinta. Esto con el fin de explotar las propiedades convexas del conjunto factible para las variables continuas y garantizar la factibilidad sobre las variables discretas. Además, a diferencia de los métodos basados en BBM para optimización mixta multiobjetivo, MOMIPGA es capaz de tratar con toda clase de variables discretas; tanto con binarias o como enteras. Para las variables binarias se realiza la cruza de un punto. Ésta consiste en elegir un punto aleatorio entre la longitud del cromosoma de los individuos padre. A partir de este punto se generan dos individuos nuevos intercambiando los cromosomas de los padres. En la Figura 3.3 se muestra un ejemplo de cruza de un punto.

Para las variables que son enteras se realiza el tipo de cruza denominada *Laplace crossover*[Deep et al., 2009]. En este tipo de cruza se tienen las primeras n_e entradas de los individuos padres $x_e^1 = (x_1^1, x_2^1, \ldots, x_{ne}^1)$ y $x_e^2 = (x_1^2, x_2^2, \ldots, x_{ne}^2)$. Los individuos hijo los denotaremos por h_e^1 y h_e^2. Primero generamos números aleatorios $u_i, r_i \in [0, 1]$ y calculamos β_i para $i = 1, \ldots, n_e$ usando (3.1), donde a y b son constantes que se sugiere determinar como $a = 0$ y $b = 0.35$.

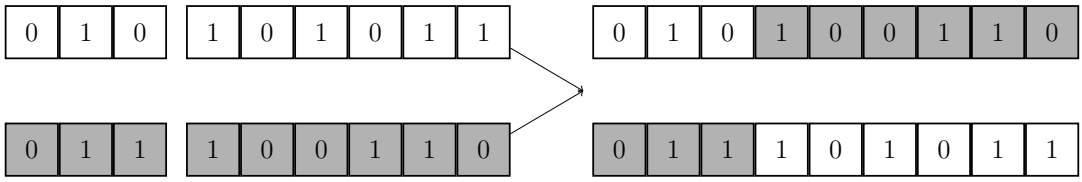

Figura 3.3: Recombinación genética de un punto para las variables binarias. Se realiza el intercambio a partir del punto 3 del cromosoma.

$$\beta_i = \begin{cases} a - b\log(u_i), & \text{si } \ r_i \leq 0.5 \\ a + b\log(u_i), & \text{si } \ r_i > 0.5 \end{cases} \tag{3.1}$$

Finalmente, después de calcular b_i, los hijos se producen de la siguiente manera

$$\begin{aligned} h_i^1 =& x_i^1 + \beta_i |x_i^1 - x_i^2| \\ h_i^2 =& x_i^2 + \beta_i |x_i^1 - x_i^2|. \end{aligned}$$

Para las variables que son continuas usamos otro tipo de técnica. Como el conjunto factible para las variables continuas es una conjunto convexo, surge de manera natural el recombinar las componentes de los individuos que son continuas como una combinación convexa. luego si x_c^1 y x_c^2 es un vector con las n_c entradas continuas del los individuos padre y denotamos por h_c^1, h_c^2 a la parte continua de los hijos entonces aplicamos la ecuación (3.2), donde α es un valor aleatorio en $[0, 1]$.

$$\begin{aligned} h_c^1 &= \alpha x_c^1 + (1-\alpha) x_c^2 \\ h_c^2 &= \alpha x_c^2 + (1-\alpha) x_c^1. \end{aligned} \tag{3.2}$$

Mutación

La mutación es el nombre que se da al operador genético que genera pequeñas variaciones en un individuo. Las variaciones son aplicadas a la representación del individuo. Así ésta altera uno o mas componentes del individuo desde su estado inicial. Es claro que la forma en que la mutación opera depende de la representación que se utilice. El operador de mutación se aplica de acuerdo a una probabilidad de ocurrencia la cual es definida por el usuario y se recomienda que sea un valor entre 0.01 y 0.1. Esto para evitar la perdida de buenos individuos con rapidez, pues si la

mutación ocurre con demasiada frecuencia ésta afectara un gran número de individuos dentro de la población.

Después del proceso de recombinación, los hijos se someten a la etapa de mutación. Mientras la fase de reproducción se ocupa de ayudar a la exploración de todo el espacio de búsqueda, la mutación explota la solución actual para encontrar otras mejores. La mutación se ve como un operador para mantener la diversidad genética en la población pues se introducen nuevas estructuras genéticas en la población mediante la modificación aleatoria de los individuos. El propósito principal de la mutación es introducir diversidad. Evita que la búsqueda se atore en un óptimo local, pues se pretende que los individuos de la población no sean idénticos.

Como ya se menciono, el operador de mutación depende del tipo de componente al que se aplica. Para las variables discretas binarias únicamente se realiza un cambio de valor, por ejemplo si la $i-$ésima variable de un individuo es $x_i = 0$, al aplicarle la mutación su valor se cambia por $x_i = 1$. Para las variables discretas enteras se cambia el valor actual por un valor aleatorio entre las cotas de la variable. Por ultimo, para las variables continuas se realiza una mutación con distribución normal. Esto quiere decir que a la varibale que se muta se le suma una valor aleatorio que resulte de una distribución normal con media cero y desviación estándar uno.

3.2.4. Manejo de restricciones y selección

Una parte fundamental del funcionamiento de un AG es, sin lugar a dudas, el proceso de selección de individuos a recombinarse. En el algoritmo genético este proceso de selección suele realizarse de forma probabilística, es decir, aún los individuos menos aptos tienen una cierta oportunidad de sobrevivir. En la literatura existen distintas técnicas para llevar el proceso de selección. En MOMIPGA se usa una selección por torneo. Ésta consiste en elegir de manera aleatoria p individuos y seleccionar sólo uno con base en comparaciones directas. Para la selección de torneo de MOMIPGA se utiliza $p = 2$, que comúnmente se conoce como torneo binario.

Dadas las restricciones del problema, los individuos pueden ser factibles o infactibles. Por lo tanto se pueden distinguir tres casos cuando se tienen dos individuos: ambos son factibles; uno es factible y otro es infactible; o ambos son infactibles. Para tratar estos tres casos y realizar las comparaciones del proceso de selección se asigna a cada individuo de la población un rango de dominancia como se describe en [Deb et al., 2002].

Primero consideremos sólo los individuos que son factibles. Con ayuda de la relación de dominancia se pueden determinan aquellos que son no dominados. Estos individuos serán los mejores, pues además de ser factibles son individuos no dominados respecto a toda la población. Por lo tanto se les asigna el rango de dominancia 1. Además como ya se les ha asignado un rango se omiten del proceso de asignación de rangos. El siguiente rango estará comprendido por los individuos factibles que resulten no dominados sin tomar en cuenta a los individuos de rango 1. El proceso se sigue hasta asignar a cada individuo factible un rango. Finalmente, a los individuos que son infactibles se les asigna el rango más bajo posible, es decir, si r es el último rango de los individuos factibles entonces $r+1$ será el rango de los individuos infactibles. En la Figura 3.4 se ilustra una población de individuos factibles clasificados de acuerdo al rango de dominancia. El frente de Pareto se muestra como referencia.

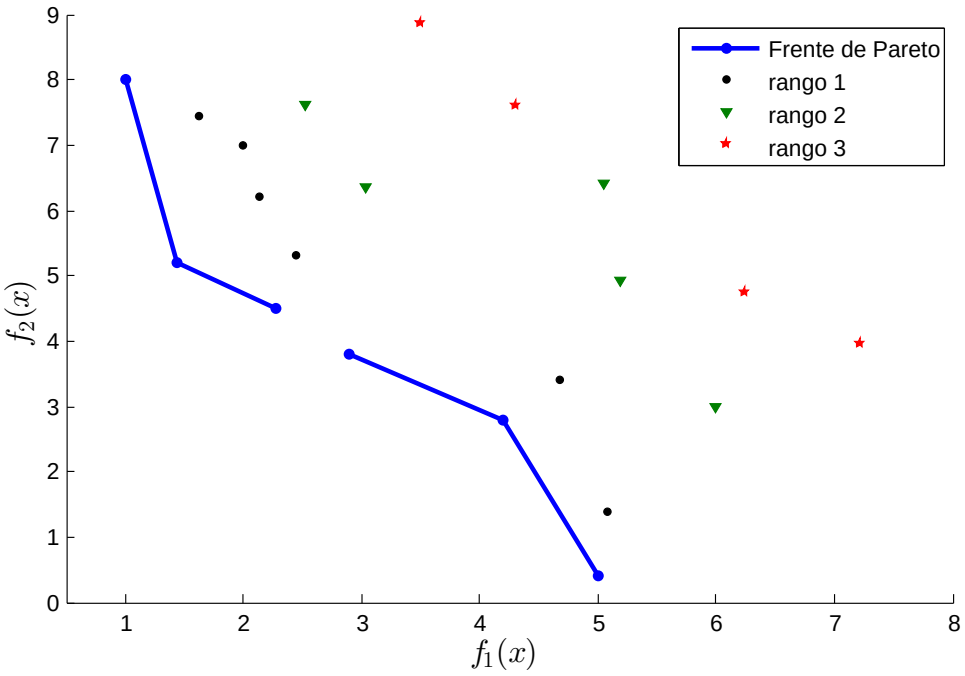

Figura 3.4: Frente de Pareto y población de individuos factibles clasificados respecto a su rango de dominancia.

Es claro que los rangos de dominancia clasifican a los individuos por subconjuntos, por lo que éstos se encuentran ordenados de acuerdo al subconjunto al que pertenecen. Es decir, siempre que dos individuos pertenezcan cada uno a distinto subconjunto se puede determinar cual es el mejor (aquel que pertenece al subconjunto de menor rango). Sin embargo, los individuos de un mismo subconjunto son incomparables entre sí. Para comparar dos individuos de un mismo rango es necesario establecer las condiciones necesarias para determinar que individuo es mejor que otro. Para estos casos, MOMIPGA usa la distancia de agrupamiento definida en [Deb et al., 2002] para comparar individuos factibles y el valor de violación de restricciones para individuos infactibles.

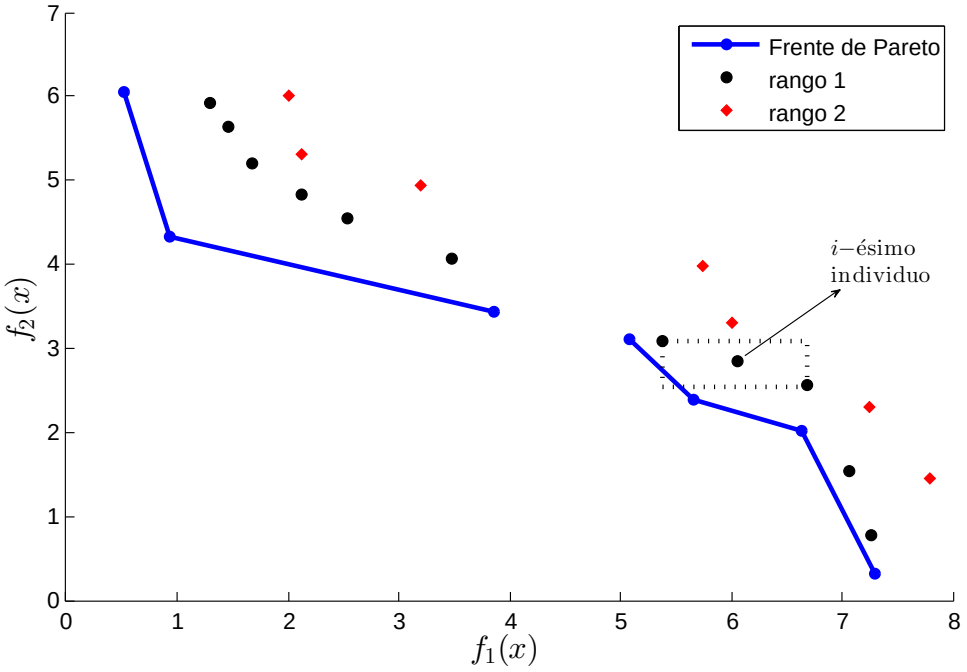

Figura 3.5: Calculo de la distancia de agrupamiento para el $i-$ésimo individuo.

La distancia de agrupamiento se utiliza para obtener una estimación de la densidad de individuos que se encuentran en los alrededores de un individuo particular en la población. El objetivo es conseguir que los individuos se encuentren distribuidos a

lo largo de todo el frente de Pareto. Por lo tanto esta distancia se estima usando el perímetro del paralelogramo formado por el uso de los dos individuos más cercanos como los vértices. Para obtenerla se calcula la distancia promedio de dos puntos a cada lado del individuo para cada uno de los objetivos. En la Figura 3.5, la distancia de agrupamiento del $i-$ésimo individuo corresponde a la longitud lateral promedio del paralelogramo marcado con lineas discontinuas. Notemos que entre más grande sea la distancia de agrupamiento el individuo correspondiente estará más aislado. Así es preferible seleccionar (para que se preserve en la población) un individuo cuyo alrededor este menos denso para así conseguir que los individuos se distribuyan en toda la extensión del frente de Pareto.

Por otra parte la violación de restricciones se calcula usando la ecuación (3.3), donde X es un individuo infactible y I_x representa el conjunto de índices de las restricciones que X no satisface. Entre mayor sea esta cantidad el individuo se encuentra mas alejado de la región factible. Así al comparar dos individuos infactibles se selecciona el que obtenga una menor violación de restricciones.

$$\sum_{i \in I_x} |A_i X - b_i| \tag{3.3}$$

En resumen, el proceso de selección compara el rango de dominancia de los individuos y elige al individuo con mejor rango (tomando en cuenta que un rango menor es mejor que una más alto). Cuando ambos individuos tienen el mismo rango se realiza un desempate. Si además de pertenecer al mismo rango los individuos son factibles, se calcula su distancia de agrupamiento y se elige aquel que tenga el mayor valor de dicha distancia. Por otra parte si los individuos son infactibles, se calcula su valor de violación de restricciones y se elige el que resulte menor. Este proceso garantiza que los individuos factibles se prefieran a los infacibles pero además que entre los individuos de un mismo rango se seleccione el mejor para preservarse en la población (el que se encuentre en una región poco explorada o el que se encuentre más cercano a la región factible).

Para hacer el cambio de generación se usa una selección +. Esta consiste en unir la población de padres e hijos y seleccionar los mejores individuos de esta unión. Por lo cual después de realizar la unión se debe asignar el rango de dominancia para cada individuo. Una vez ordenados por su rango la siguiente población se va construyendo con los mejores individuos. Sin embargo hay que tener en mente que el tamaño de población se mantiene constante, es decir, el número de individuos que se procesan en cada iteración es el mismo. Entonces la población se construye agregando individuos

rango por rango. Cuando al agregar los individuos de un rango a la población el tamaño de población se rebasa, solo se agregan los individuos necesarios y de acuerdo a su distancia de agrupamiento (si son factibles) ó a la violación de restricción (en caso de ser infactibles).

3.3. Método híbrido con MOMIPGA

Como se sabe los AGs son algoritmos de aproximación. El resultado de una ejecución es una aproximación al resultado buscado. En particular, el algoritmo MOMIPGA devuelve una aproximación del frente de Pareto para el problema dado. Por ejemplo, en la Figura 3.6 se muestra el frente de Pareto real y una aproximación obtenida con MOMIPGA. Como se observa el resultado obtenido con MOMIPGA se encuentra alejado del frente de Pareto real. Sin embargo, para que MOMIPGA sea competitivo con los demás algoritmos para MOMIP es necesario que éste pueda generar el frente de Pareto con certeza. Es decir, el algoritmo debe ser capaz de generar soluciones sobre el frente de Pareto.

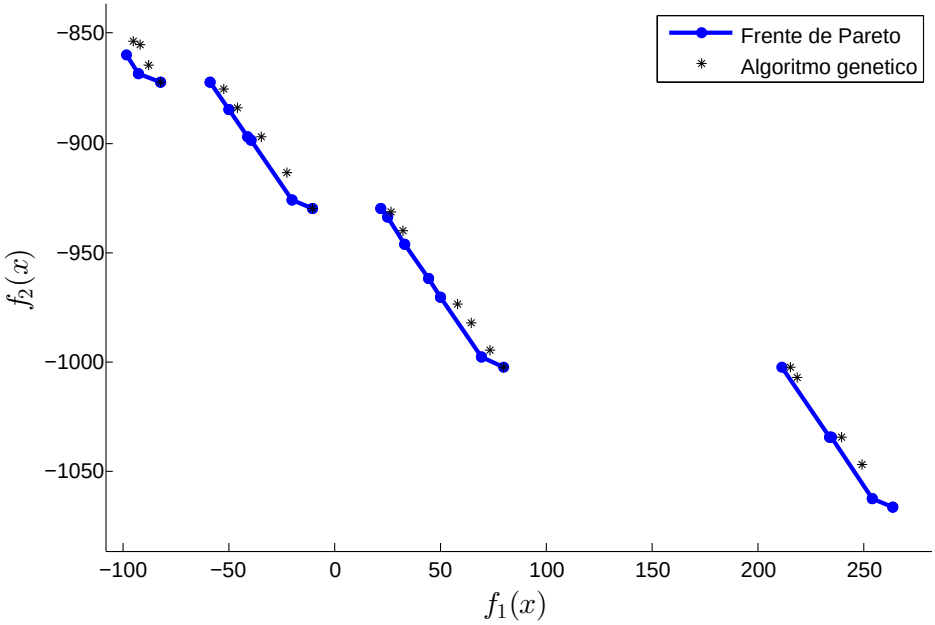

Figura 3.6: Resultado de una ejecución del algoritmo MOMIPGA y frente de Pareto real.

Durante el análisis del algoritmo MOMIPGA se encontró que éste tenia la ventaja de llegar con cierta rapidez a las soluciones de las variables discretas, sin embargo para las variables continuas era muy costoso determinarlas. Analizando la imagen anterior se puede observar este hecho. En efecto, se muestra como las soluciones están próximas a cada parte del frente de Pareto pero no llegan a él por completo.

El algoritmo MOMIPGA determina las combinaciones sobre las variables discretas. Durante las ejecuciones del algoritmo se encontró que los individuos cercanos a cierta parte del frente de Pareto tenían los mismo valores en las componentes discretas. Como se explica en la Sección 2.2.1 una vez que las combinaciones en las variables discretas se han determinado se puede aplicar a cada combinación el método Simplex Multiobjetivo para obtener las variables continuas y así determinar el frente de Pareto. Basados en este resultado, el algoritmo MOMIPGA se complementó con el método Simplex Multiobjetivo. Después de realizar cierto número de iteraciones (definido por el usuario) MOMIPGA selecciona las componentes discretas y para cada una de ellas se ejecuta el Simplex Multiobjetivo. Así el algoritmo devuelve los puntos extremos del frente de Pareto. Con ellos se puede construir todo el frente al calcular combinaciones convexas de dos puntos consecutivos. Por lo que MOMIPGA devuelve los óptimos de Pareto, sus correspondientes puntos del frente de Pareto y un indicador binario que determina si la siguiente solución esta conectada o no.

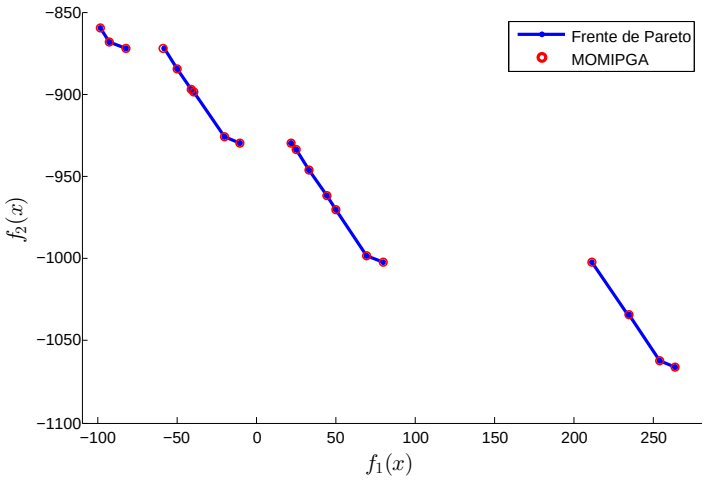

Figura 3.7: Resultado de una ejecución del algoritmo MOMIPGA con el uso del Simplex Multiobjetivo y frente de Pareto real.

La Figura 3.7 muestra el resultado de la ejecución de MOMIPGA con el uso del método Simplex Multiobjetivo. Como se explica ahora el resultado corresponde a los puntos extremos del frente de Pareto. Con esta mejora el algoritmo devuelve soluciones sobre el frente de Pareto y no sólo aproximadas. Sin embargo cuando MOMIPGA no es capaz obtener todas las combinaciones de variables discretas que forman el frente de Pareto, el método Simplex no considera estas combinaciones y por lo tanto el algoritmo devuelve una solución parcial y no el frente completo. En la Figura 3.8, por ejemplo, se observa enmarcado con lineas discontinuas dos secciones del frente de Pareto que MOMIPGA no puede generar.

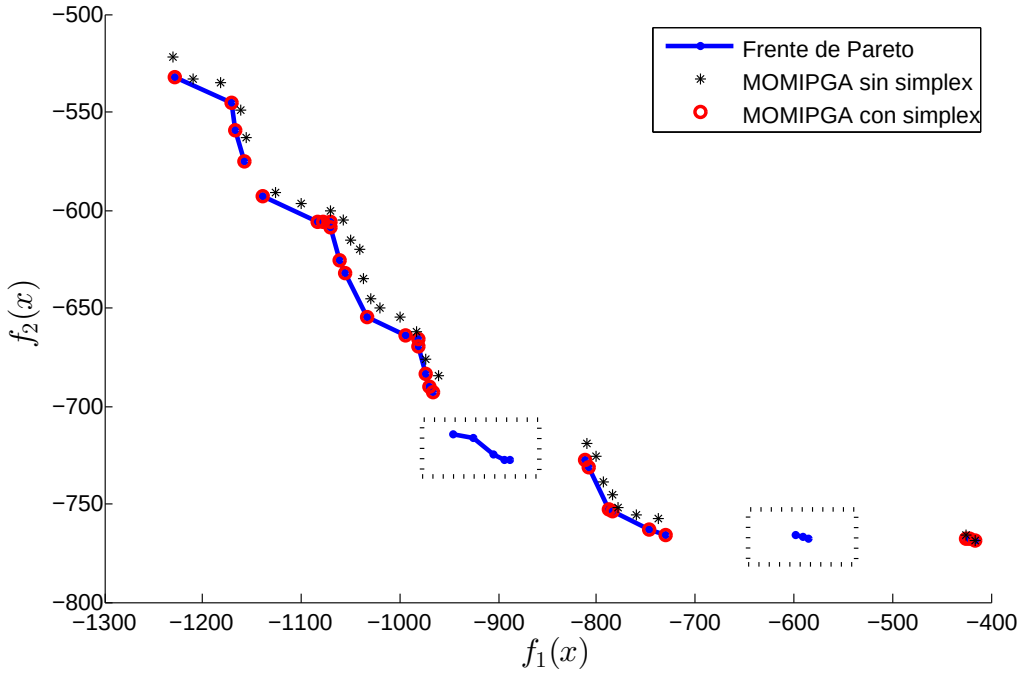

Figura 3.8: Resultado de una ejecución del algoritmo MOMIPGA con y sin el método Simplex Multiobjetivo y frente de Pareto real.

Para resolver estos casos se incorporó a MOMIPGA el método TSM. La idea es utilizar el TSM en cada sección intermedia del frente de Pareto obtenida con MOMIPGA. Así después de realizar las ejecuciones típicas del algoritmo genético y aplicar el método Simplex Multiobjetivo, MOMIPGA ejecuta el método TSM donde se detecta un cambio de sección del frente de Pareto. Para su ejecución se determina

que la búsqueda se inicie en el rectángulo formado por el último punto una sección y el primer punto de la sección subsecuente. En la Figura 3.9 se muestra la ejecución de MOMIPGA usando el Simplex Multiobjetivo (para acercar los individuos al frente de Pareto) y el método TSM para obtener regiones faltantes del frente. Notemos que en este caso se obtiene por completo los puntos extremos que construyen el frente de Pareto. Asimismo se define a MOMIPGA como un algoritmo híbrido pues además de contar con un algoritmo genético se incorpora el uso del Simplex Multiobjetivo y el TSM.

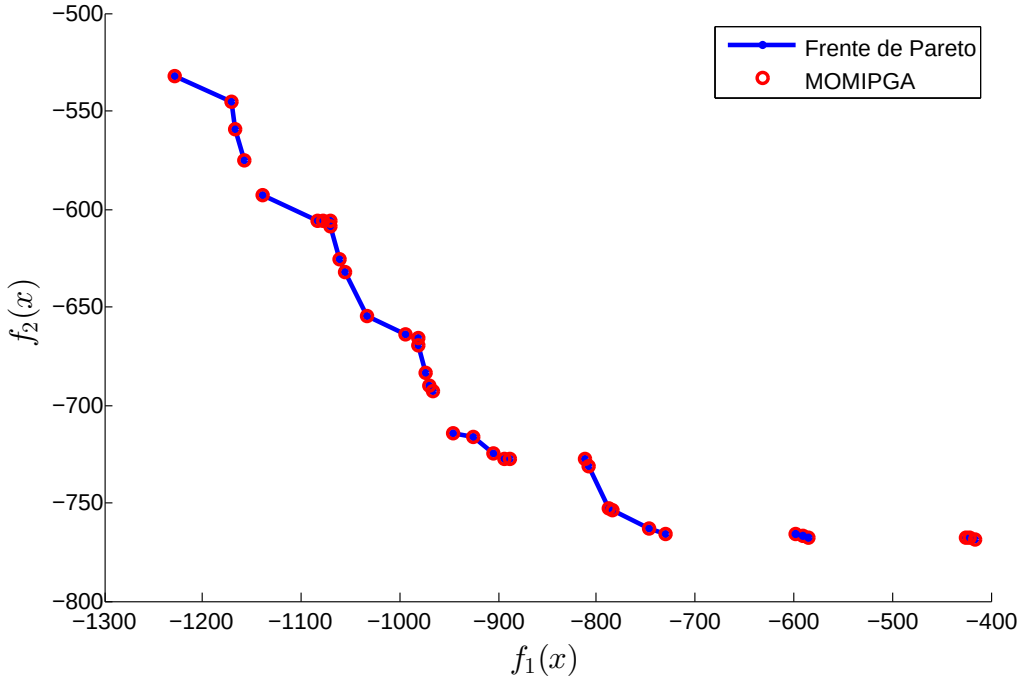

Figura 3.9: Frente de Pareto y resultado obtenido de una ejecución del algoritmo MOMIPGA usando Simplex Multiobjetivo y TSM.

En el Algoritmo 3.3.1 se presenta un pseudocódigo del algoritmo híbrido propuesto, el cual como ya se mencionó lo denominados MOMIPGA. En la primer etapa se ejecuta un algoritmo genético cuyo operadores fueron diseñados e incorporados específicamente para resolver problemas lineales multiobjetivo con variables discretas. Al inicio del algoritmo se inicializan las soluciones o, en términos del algoritmo los individuos, con ayuda de las soluciones obtenidos del Simplex Multiobjetivo. En la etapa final se ejecuta el métodos TSM en las secciones del frente que son disconti-

nuas para determinar si en efecto existe la discontinuidad u obtener soluciones que el algoritmo genético no logro alcanzar.

Algoritmo 3.3.1 Descripción del algoritmo MOMIPGA

Entrada: Datos del problema multiobjetivo. Parámetros del algoritmo genético
$n-$Número de generaciones.
$m-$Número de individuos.
ρ_c-probabilidad de cruza.
ρ_m-probabilidad de mutación.
Salida: Conjunto de puntos no dominados pertenecientes a $\mathcal{P}_s$ y $\mathcal{P}_f$.

1: Resolver problema multiobjetivo sin restricciones de integridad, uso de Simplex Multiobjetivo.
I es el conjunto de individuos actual.
2: **for each** individuo **do**
3: Inicializar a partir de las soluciones obtenidas en paso 1.
4: Verificar factibilidad y corregir variables discretas ▷ Sección 3.2.2.
5: Agregar al conjunto I.
6: **end for**
7: $i \leftarrow 0$
H es el conjunto de individuos hijos.
8: **while** $i < n$ **do**
9: $H \leftarrow \emptyset$ ▷ H se inicializa como el conjunto vacío.
10: **for** $j \leftarrow 1$ **to** $\frac{m}{2}$ **do**
11: Seleccionar dos individuos de I,digamos p_1 y p_2 ▷ Sección 3.2.4.
12: Generar un número aleatorio r entre 0 y 1.
13: **if** $\rho_c \geq r$ **then**
14: Se generan los individuos h_j y $h_{j+m/2}$.
a partir de p_1 y p_2. ▷ Ver técnica de cruza, Sección 3.2.3.
15: **else**
16: $h_j \leftarrow p_1$
17: $h_{j+m/2} \leftarrow p_2$
18: **end if**
19: **if** $\rho_m \geq r$ **then**
20: Realizar mutación sobre h_j y $h_{j+m/2}$. ▷ Ver Mutación, Sección 3.2.3.
21: **end if**

Continúa en la siguiente página.

Continuación del Algoritmo 3.3.1.

Verificar factibilidad de los individuos h_j y $h_{j+m/2}$.
Agregar h_j y $h_{j+m/2}$ al conjunto H.
end for
$Aux = I \bigcup H$ ▷ Aux es el conjunto unión de I y H.
Seleccionar de Aux los mejores m individuos. ▷ Sección 3.2.4.
$I \leftarrow Aux$ ▷ Reemplazar el conjunto I.
end while
Se determinan las secciones discontinuas del frente, analizando las variables discretas de los individuos de rango 1.
for each sección **do**
Se ejecuta el método TSM. ▷ Hibridación con TSM, Sección 3.3.
end for

Capítulo 4

Resultados y discusión

En este capítulo se muestran los resultados numéricos de la aplicación del algoritmo propuesto MOMIPGA sobre problemas de prueba. Se evalúa la eficacia del método para generar soluciones sobre el frente de Pareto. De igual manera se muestra la comparación con respecto al método TSM en términos de:

1. El número de vértices calculados sobre $\mathcal{P}_f$,
2. El costo computacional en términos del número de evaluaciones de la función objetivo requeridos para su cálculo, y
3. El tiempo de ejecución total de los métodos.

Para el estudio experimental que se llevo a cabo se usaron problemas de tipo académico para los cuales ya se conoce el frente de Pareto. Estos problemas forman parte de la biblioteca **Biobjective Mixed Integer Programs Instances** de la Universidad de New Castle y se pueden encontrar en la siguiente dirección

`http://ogma.newcastle.edu.au:8080/vital/access/manager/Repository/uon:13218`

La biblioteca utilizada cuenta con 3 diferentes clases de problemas, las cuales se describen en la tabla 4.1. Además para cada clase se tienen 5 instancias que solo varían en las funciones objetivo y la matriz de restricciones, por lo que en general su única diferencia es la forma del frente de Pareto.

Este tipo de problemas se generaron de forma aleatoria usando los mismos intervalos que Mavrotas y Diakoulaki en su articulo [Mavrotas and Diakoulaki, 1998],

Nombre	Número de restricciones	Número de variables	Variables binarias	Variables enteras
C20	20	20	10	10
C40	40	40	20	20
C80	80	80	40	40

Tabla 4.1: Descripción de los problemas usados para el estudio experimental.

donde además se consideran las variables discretas como binarias. Hay que destacar que esta clase de problemas es la más utilizada en los estudios computaciones de los algoritmos conocidos para MOMIPs. A continuación se dan los intervalos utilizados para cada elemento del problema:

- $[-10, 10]$ para los coeficientes de las variables continuas en las funciones objetivo.
- $[-200, 200]$ para los coeficientes de las variables discretas en las funciones objetivo.
- $[50, 100]$ para los coeficientes que forman la matriz de restricciones (matriz A).
- $[-1, 20]$ para los coeficientes que forman el vector de restricciones (vector b).

Cada variable binaria se incluye únicamente en una restricción con un coeficiente aleatorio. Además la suma de todas las variables binarias se restringe a ser menor o igual a $n_e/3$ (n_e es el número de variables discretas). Asimismo, la matriz de restricciones es una matriz dispersa al 75 %, es decir, aproximadamente 75 % de los elementos de la matriz A son cero.

Para cada instancia del problema de prueba se realizaron 30 ejecuciones del algoritmo MOMIPGA durante 100 generaciones con una población de 100 individuos. La probabilidad de cruza se estableció en $\rho_c = 0{,}9$ y la probabilidad de mutación en $\rho_m = 0{,}1$. Al termino de las ejecuciones se obtuvo el peor valor, el mejor valor, la media y la desviación estándar de los siguientes datos: puntos extremos obtenidos, número de llamadas a la función, tiempo de ejecución, distancia generacional y distancia generacional invertida. Para cada problema se muestra una tabla comparativa del algoritmo MOMIPGA, donde los valores obtenidos para este algoritmo se encuentran organizados de la siguiente manera: la primer columna corresponde al peor valor obtenido, mientras que la segunda y tercera son los mejores valores y el

promedio, respectivamente. Asimismo, enseguida del promedio y entre paréntesis se muestra la desviación estándar.

Para poder determinar la eficacia del algoritmo desarrollado es necesario contar con un criterio que permita evaluar de manera objetiva y cuantitativa el desempeño del mismo. Esta necesidad ha traído consigo el desarrollo de una serie de indicadores de desempeño que evalúan, entre otras cosas, la precisión y la calidad de los resultados obtenidos. Tal es el caso de los indicadores denominados *distancia generacional*, *distancia generacional invertida*.

Cada indicador trata de describir de forma cuantitativa algún aspecto de interés en una aproximación a un $\mathcal{P}_f$ dado y para lograrlo se realizan diversas operaciones con los elementos de la aproximación que se tiene y una discretización del frente real. Una característica importante para usar estos indicadores es el conocimiento previo del verdadero frente de Pareto para el problema dado. Así denotamos por PF_r al frente verdadero mientras que PF_a es una aproximación dada de PF_r.

La *distancia generacional* (DG) es un indicador de desempeño que nos indica que tan lejos, en promedio, se encuentra FP_{real} de FP_a [Coello et al., 2002]. Matemáticamente la distancia generacional esta definida como

Definición 4.1. Sea FP_r el frente de Pareto verdadero y FP_a una aproximación de FP_r con $|FP_a| = n$. La distancia generacional (DG) entre FP_r y FP_a esta dada por

$$DG = \frac{\left(\sum_{i=1}^{n} d_i^p\right)^{1/p}}{n}$$

donde d_i es la distancia euclidiana entre cada elemento, i, de FP_a y el elemento más cercano de FP_r a tal miembro.

La finalidad de la distancia generacional es medir la cercanía de una aproximación con el frente verdadero. Notemos que $DG = 0$ cuando $FP_a = FP_r$, por lo que valores cercanos a cero indicarían que FP_a es una aproximación bastante cercana a FP_r. Sin embargo la forma en que se calcula la proximidad al frente con este indicador presenta una desventaja importante, pues esta medida no considera la dispersión de los puntos de FP_a sobre FP_r. Siempre que los elementos de FP_a se encuentren cerca o sobre el FP_r se obtendrá un buen valor. Por ejemplo, si FP_a contiene sólo un punto y éste se encuentra sobre FP_r, se obtendría el valor de $DG = 0$, sin embargo

un punto no es representativo de un conjunto como FP_r.

Con el fin de evaluar la extensión de los puntos obtenidos por la aproximación sobre el frente de Pareto real, se utiliza la distancia generacional invertida (DGI). Este indicador de desempeño es similar a la distancia generacional, solo que aquí las distancias ahora son medidas de cada uno de los elementos de FP_r hacia el elemento más cercano de FP_a [Coello and Cortés, 2005]. Formalmente la distancia generacional invertida esta definida por

Definición 4.2. Sea FP_r el frente de Pareto verdadero y FP_a una aproximación de FP_r con $|FP_r| = m$. La distancia generacional invertida (DGI) esta dada por

$$DGI = \frac{\left(\sum_{j=1}^{m} d_j^p\right)^{1/p}}{m}$$

donde d_j es la distancia euclidiana entre cada elemento, j, de FP_r y el elemento más cercano a él en FP_a.

Al medir las distancias de cada elemento de FP_r al elemento más cercano de PF_a es posible identificar si los puntos de la aproximación se encuentran distribuidos sobre todo el frente de Pareto real. Retomando el ejemplo anterior, en el cual FP_a contiene unicamente un punto, podemos notar que el valor de DGI sería muy grande. Al considerar solo un punto, las distancias que se promedian serán las que hay entre este punto y cada uno de los elementos del FP_r, así entre mas alejados estén estos elementos de nuestro punto, estas distancias serán mayores y en consecuencia también el valor del indicador DGI. Cuando FP_a no tiene una buena extensión sobre FP_r, el valor de DGI será grande.

Recientemente se propuso un nuevo indicador que adapta la distancia de Hausdorff para comparar frentes de Pareto. Este indicador puede medir el desempeño de un algoritmo de búsqueda estocástico, tal como nuestro algoritmo híbrido MOMIPGA, pues esta compuesto por ciertas modificaciones de los indicadores DG y DGI. En lugar de calcular para cada indicador la media de las distancias, ahora se calcula la media generalizada o también conocida como media de Hören. Así pues los indicadores quedan descritos por las siguientes ecuaciones[Schütze et al., 2012]

$$DG_p = \left(\frac{1}{m} \sum_{j=1}^{m} d_j^p \right)^{1/p} \tag{4.1}$$

$$DGI_p = \left(\frac{1}{n} \sum_{i=1}^{n} d_i^p \right)^{1/p} \tag{4.2}$$

donde, de nuevo, d_i es la distancia euclidiana entre cada elemento, i, de FP_a y el elemento más cercano de FP_r y d_j es la distancia euclidiana entre cada elemento, j, de FP_r y el elemento más cercano a él en FP_a.

Definición 4.3. Sean $FP_r,\ FP_a \subset \mathbb{R}^k$ con $|FP_a| = n$ y $|FP_r| = m$. Se define el indicador Δ_p por

$$\Delta_p := max(DG_p, DGI_p) = max\left(\left(\frac{1}{m} \sum_{j=1}^{m} d_j^p \right)^{1/p}, \left(\frac{1}{n} \sum_{i=1}^{n} d_i^p \right)^{1/p} \right).$$

En las tablas comparativas, además de los puntos extremos se muestra también el total de puntos que cada método genera. Así por ejemplo, para la tabla del problema 20-1 el método TSM-Lp_solve obtiene 87 puntos de los cuales sólo 26 son extremos, mientras que MOMIPGA obtiene 27 puntos totales y 26 extremos. Por otro lado, es importante destacar que el algoritmo MOMIPGA es capaz de aproximar todos los puntos extremos del frente de Pareto, sin embargo en las tablas comparativas se reportan aquellos puntos que se obtienen con exactitud. Es por esto que a pesar de obtener buenos valores en los indicadores DG y DGI, los puntos extremos reportados son menores.

Como se mencionó, los métodos se probaron utilizando problemas académicos de los cuales se conoce la solución, es decir, el frente de Pareto. A continuación se muestra la gráfica del frente de Pareto real y el frente obtenido por el algoritmo MOMIPGA así como una tabla con los resultados para cada problema. Asimismo se muestran los resultados que se obtienen al ejecutar el método TSM usando dos distintas paqueterías de optimización lineal mixta como son `lp_solve`® y `Gurobi`® `Optimizer`. Para hacer la comparación entre los 3 diferentes métodos se hizo el cálculo del costo promedio por punto, que no es más que el número promedio de llamadas a la función que realiza cada método para obtener un punto del frente de Pareto, sea o no punto extremo.

Problema 20-1

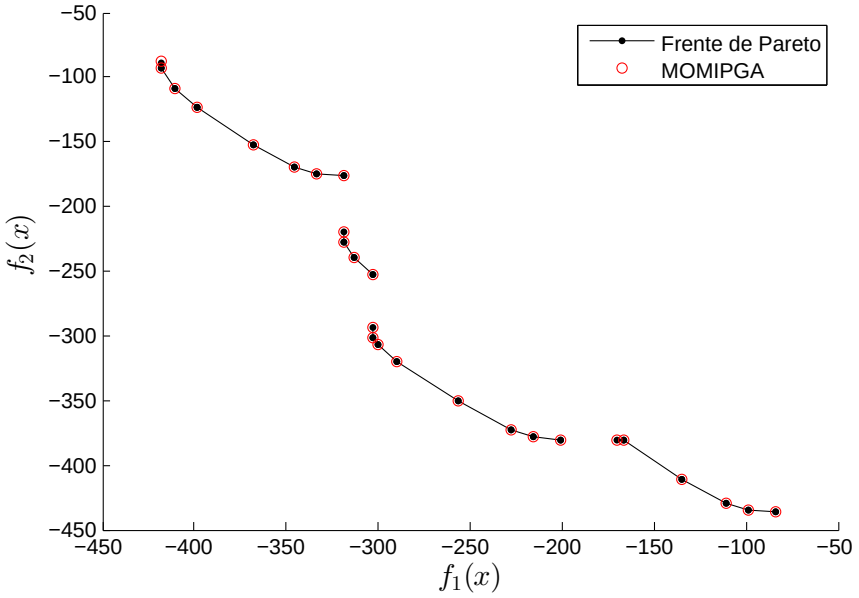

Figura 4.1: Frente de Pareto real y aproximado para el Problema C20-1

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	26	26	**26** (0)	**26**	**26**
P. obtenidos	27	27	27 (0)	35	35
# llamadas a función	967	479	**837.86** (98.64)	7,415	3,627
Costo medio por punto	35.81	17.74	**31.03**	211.85	103.62
Tiempo	18.20	16.99	17.25 (0.22)	**10.73**	11.24
Δ_2	0.0014	0.0014	0.0014 (0)		

Tabla 4.2: Resultados de los métodos MOMIPGA y TSM para el Problema C20-1.

Problema 20-2

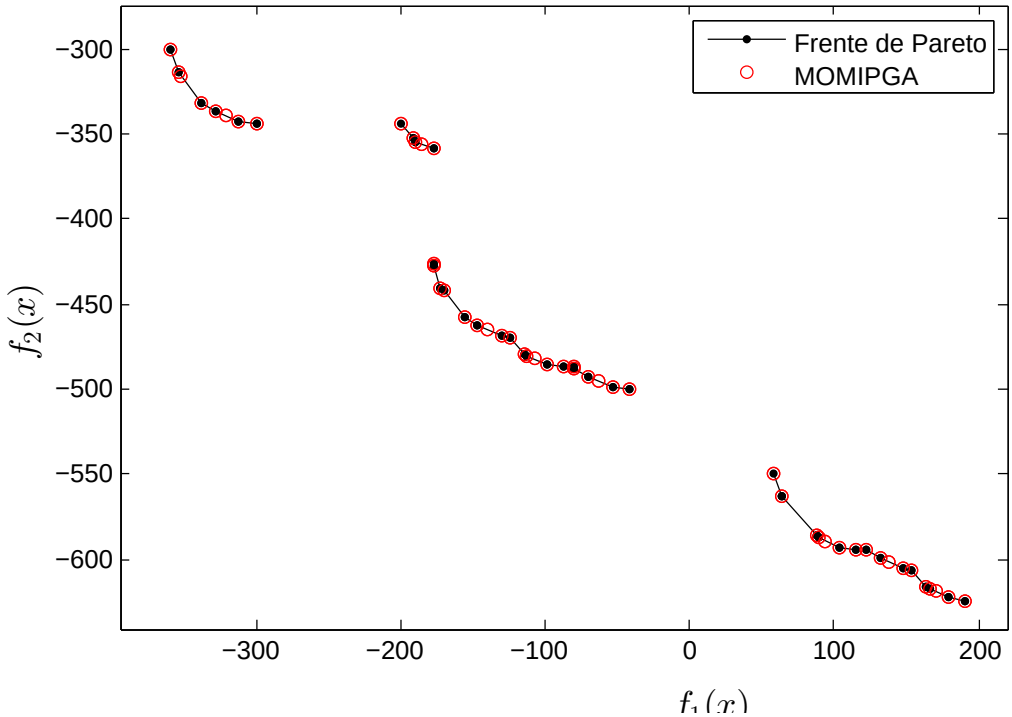

Figura 4.2: Frente de Pareto real y aproximado para el Problema C20-2

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	53	53	**53** (0)	**53**	**53**
P. obtenidos	64	64	64 (0)	78	78
# llamadas a función	8,866	5,420	**6,771.3** (952.23)	35,870	12,172
Costo medio por punto	138.53	84.68	**105.80**	459.87	156.05
Tiempo	17.10	16.65	16.88 (0.11)	**11.29**	11.93
Δ_2	0.0023	0.0023	0.0023 (0)		

Tabla 4.3: Resultados de los métodos MOMIPGA y TSM para el Problema C20-2.

Problema 20-3

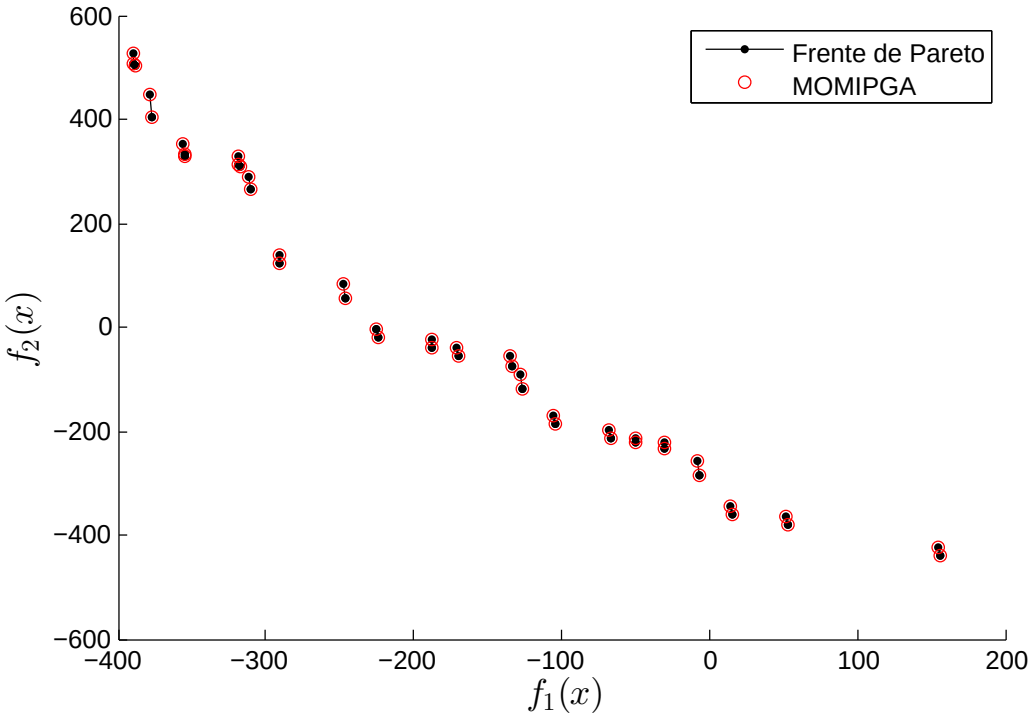

Figura 4.3: Frente de Pareto real y aproximado para el Problema C20-3

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	43	43	**43** (0)	37	43
P. obtenidos	43	43	43 (0)	38	47
# llamadas a función	8,396	5,949	**6,680.9** (484.69)	10,328	7,360
Costo medio por punto	195.25	138.34	**155.36**	271.78	156.59
Tiempo	16.13	15.70	15.88 (0.11)	**10.37**	11.22
Δ_2	0.0012	0.0012	0.0012 (0)		

Tabla 4.4: Resultados de los métodos MOMIPGA y TSM para el Problema C20-3.

Problema 20-4

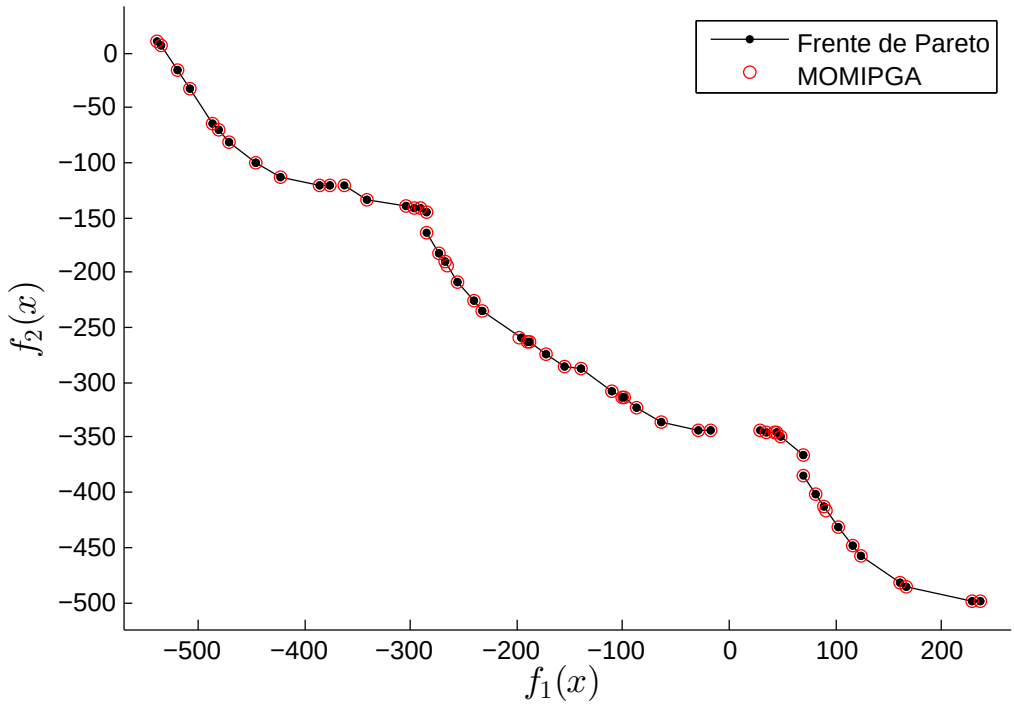

Figura 4.4: Frente de Pareto real y aproximado para el Problema C20-4

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	58	58	**58** (0)	**58**	**58**
P. obtenidos	61	61	61 (0)	85	85
# llamadas a función	9,107	6,061	**7,399.9** (826.14)	37,021	39,765
Costo medio por punto	149.29	99.36	**121.30**	435.54	467.82
Tiempo	17.11	16.54	16.82 (0.15)	**11.67**	16
Δ_2	0.0019	0.0019	0.0019 (0)		

Tabla 4.5: Resultados de los métodos MOMIPGA y TSM para el Problema C20-4.

Problema 20-5

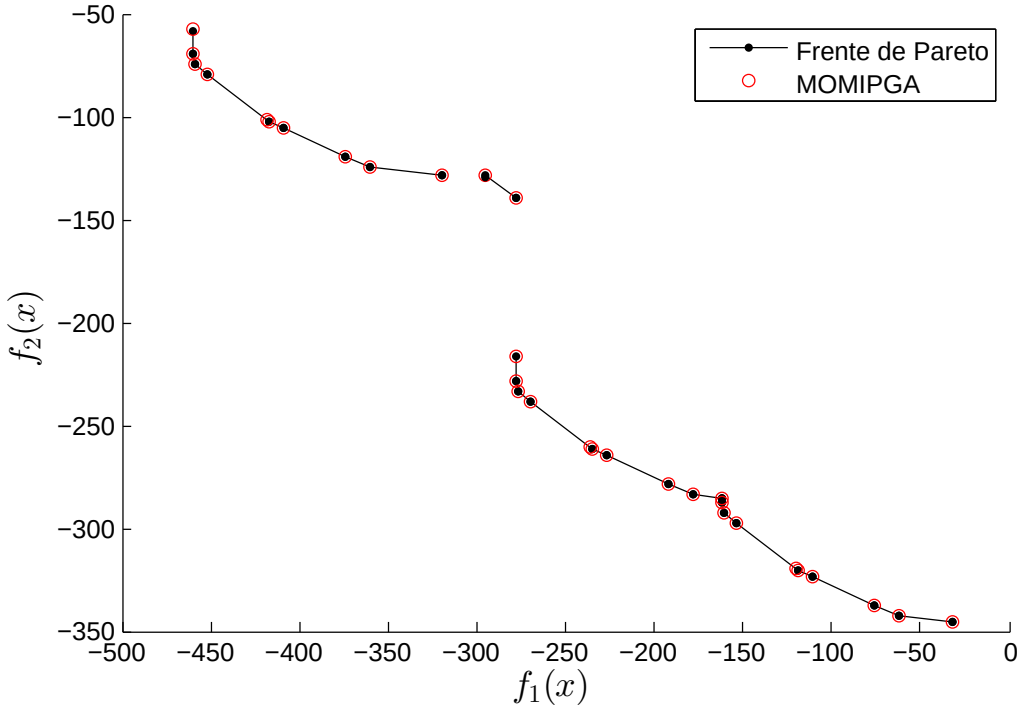

Figura 4.5: Frente de Pareto real y aproximado para el Problema C20-5

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	29	29	**29** (0)	16	25
P. obtenidos	31	31	31(0)	26	39
# llamadas a función	2,216	959	**1,643.6** (368.23)	3,086	5,279
Costo medio por punto	71.48	30.93	**53.01**	118.69	135.35
Tiempo	20.24	19.54	19.78 (0.13)	**10.49**	11.03
Δ_2	0.0020	0.0020	0.0020 (0)		

Tabla 4.6: Resultados de los métodos MOMIPGA y TSM para el Problema C20-5.

Problema 40-6

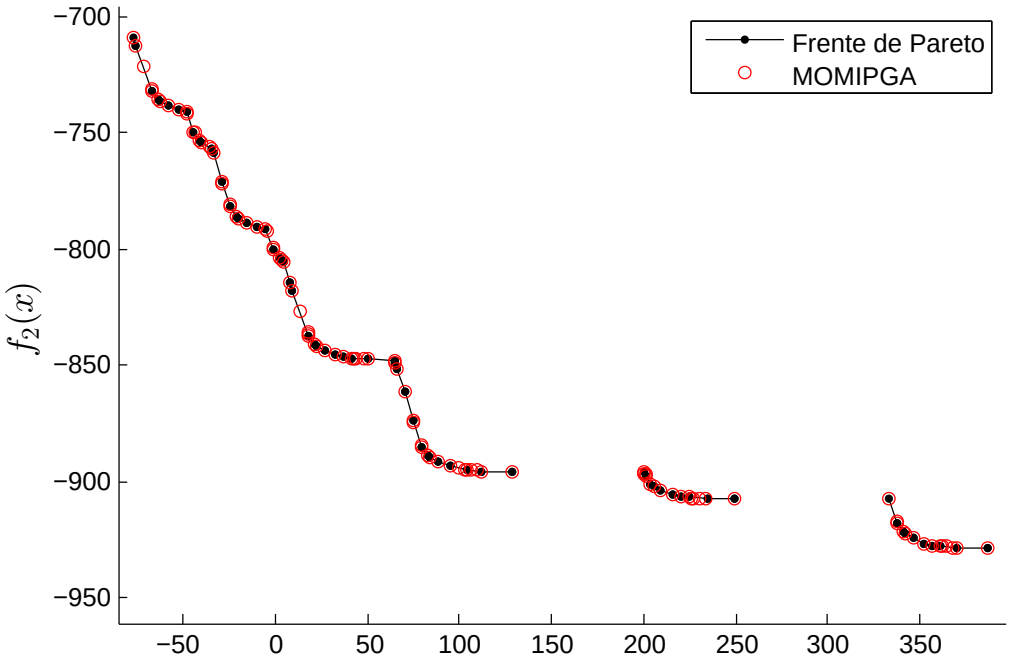

Figura 4.6: Frente de Pareto real y aproximado para el Problema C40-6

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	242	252	248.36 (3.22)	**250**	**250**
P. obtenidos	351	355	353.96 (1.29)	441	450
# llamadas a función	40,313	31,703	**36,252.20** (2,258.24)	778,448	225,654
Costo medio por punto	114.85	89.30	**102.41**	1,765.18	501.45
Tiempo	23.03	20.65	**21.22** (0.46)	34.45	36.26
Δ_2	0.0147	0.0027	0.0070 (0.0040)		

Tabla 4.7: Resultados de los métodos MOMIPGA y TSM para el Problema C40-6.

Problema 40-7

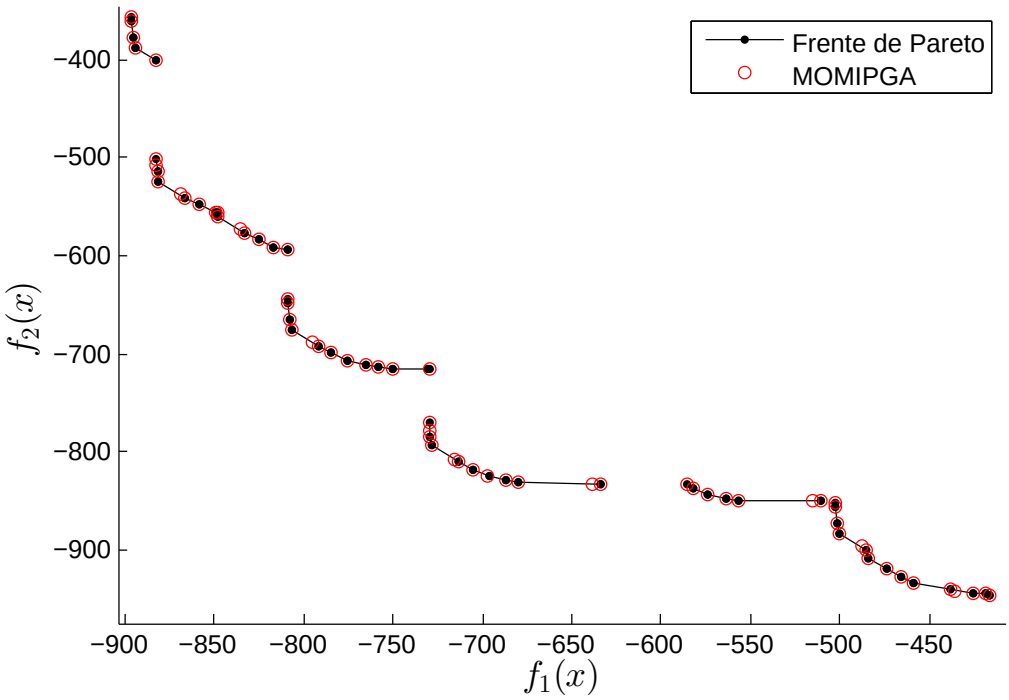

Figura 4.7: Frente de Pareto real y aproximado para el Problema C40-7

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	95	95	**95** (0)	**95**	**95**
P. obtenidos	112	112	112 (0)	139	144
# llamadas a función	17,480	12,988	**15,268.86** (1,128.65)	212,238	36,695
Costo medio por punto	156.07	115.96	**136.32**	1,526.89	254.82
Tiempo	18.04	17.39	**17.74** (0.17)	18.42	18.02
Δ_2	0.0021	0.0021	0.0021 (0)		

Tabla 4.8: Resultados de los métodos MOMIPGA y TSM para el Problema C40-7.

Problema 40-8

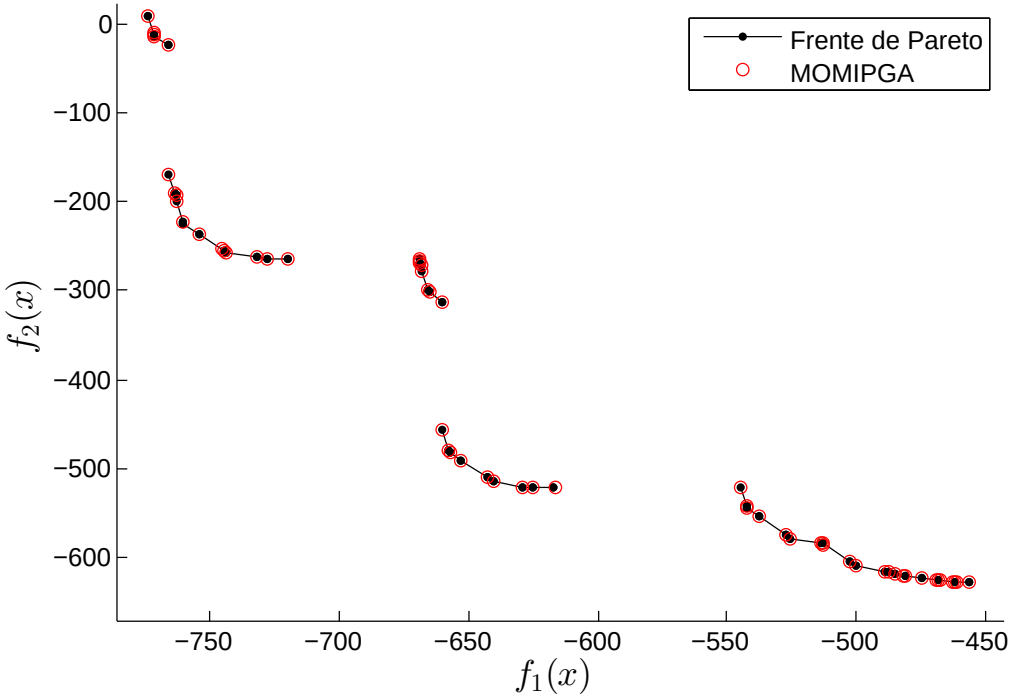

Figura 4.8: Frente de Pareto real y aproximado para el Problema C40-8

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	107	112	110.16 (2.45)	**111**	**111**
P. obtenidos	131	137	133.46 (2.95)	164	166
# llamadas a función	22,810	16,522	**19,823.26** (1,764.83)	257,274	116,675
Costo medio por punto	174.12	120.59	**148.53**	1,568.74	702.86
Tiempo	18.30	17.38	17.80 (0.26)	**14.96**	20.22
Δ_2	0.0068	0.0017	0.0047 (0.0026)		

Tabla 4.9: Resultados de los métodos MOMIPGA y TSM para el Problema C40-8.

Problema 40-9

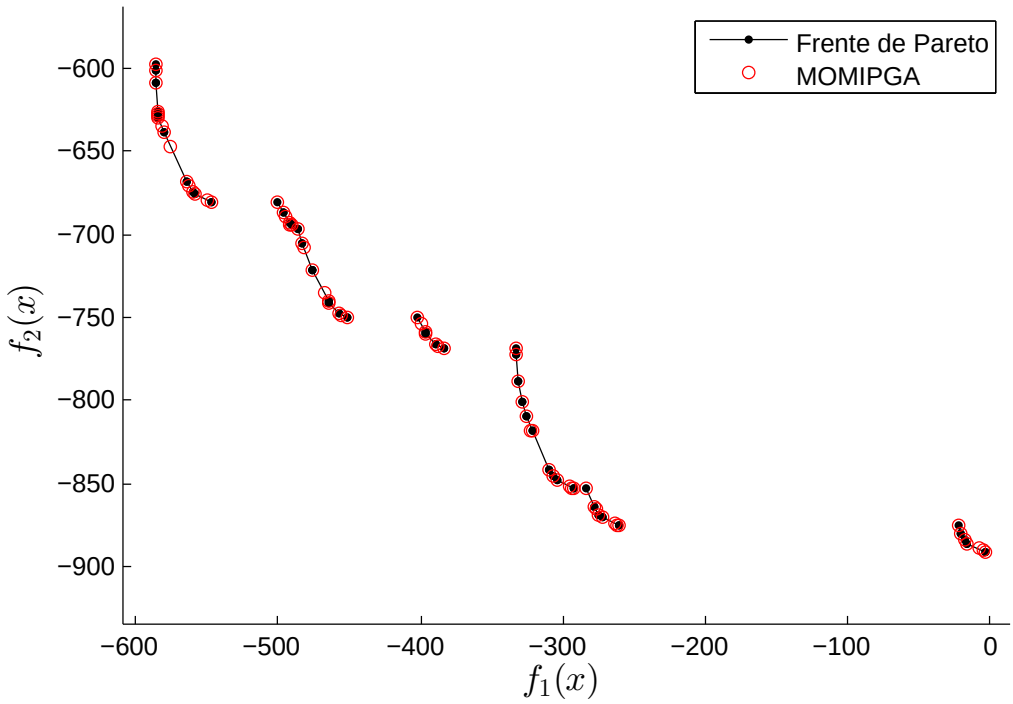

Figura 4.9: Frente de Pareto real y aproximado para el Problema C40-9

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	100	100	**100** (0)	**100**	**100**
P. obtenidos	192	192	192 (0)	210	210
# llamadas a función	21,946	14,250	**17,923.63** (2,092.78)	513,727	72,272
Costo medio por punto	114.30	74.21	**93.35**	2,446.31	344.15
Tiempo	17.46	16.35	16.86 (0.32)	15.52	**12.61**
Δ_2	0.0022	0.0022	0.0022 (0)		

Tabla 4.10: Resultados de los métodos MOMIPGA y TSM para el Problema C40-9.

Problema 40-10

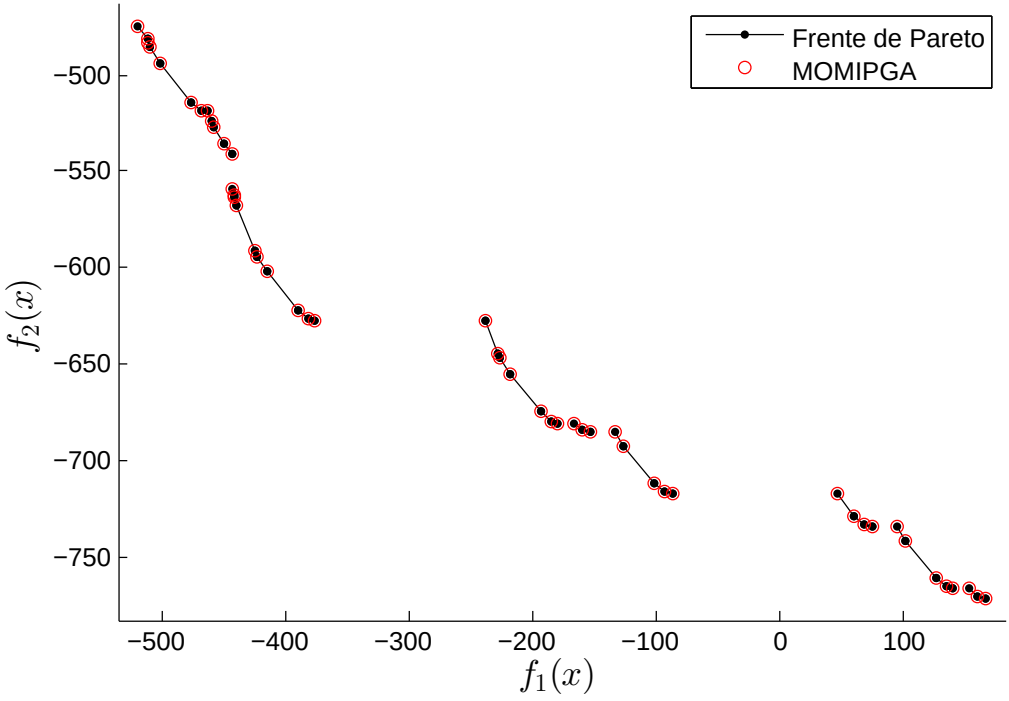

Figura 4.10: Frente de Pareto real y aproximado para el Problema C40-10

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	107	107	**107** (0)	106	102
P. obtenidos	124	124	124 (0)	155	155
# llamadas a función	19,235	15,432	**16,899.6** (1,053.64)	398,700	49,397
Costo medio por punto	155.12	124.45	**136.28**	2,572.25	318.69
Tiempo	19.20	18.38	18.73 (0.21)	**12.91**	19.26
Δ_2	0.0016	0.0016	0.0016 (0)		

Tabla 4.11: Resultados de los métodos MOMIPGA y TSM para el Problema C40-10.

Problema 80-11

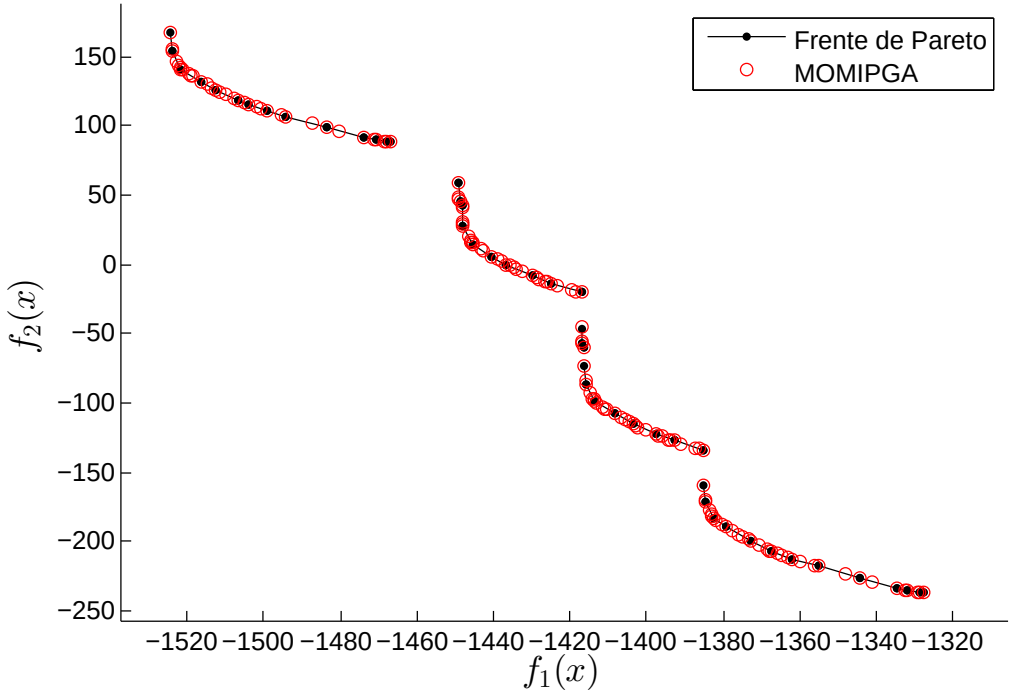

Figura 4.11: Frente de Pareto real y aproximado para el Problema C80-11

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	292	296	**293.63** (1.18)	290	292
P. obtenidos	1,038	1,079	1,072.36 (7.71)	1,181	1,303
# llamadas a función	144,100	115,292	**127,015.80** (8,555.03)	4,101,278	1,149,413
Costo medio por punto	138.82	106.85	**118.44**	3,472.71	882.12
Tiempo	55.26	48.01	**50.94** (1.95)	248.78	254.52
Δ_2	0.0597	0.0055	0.0086 (0.0109)		

Tabla 4.12: Resultados de los métodos MOMIPGA y TSM para el Problema C80-11.

Problema 80-12

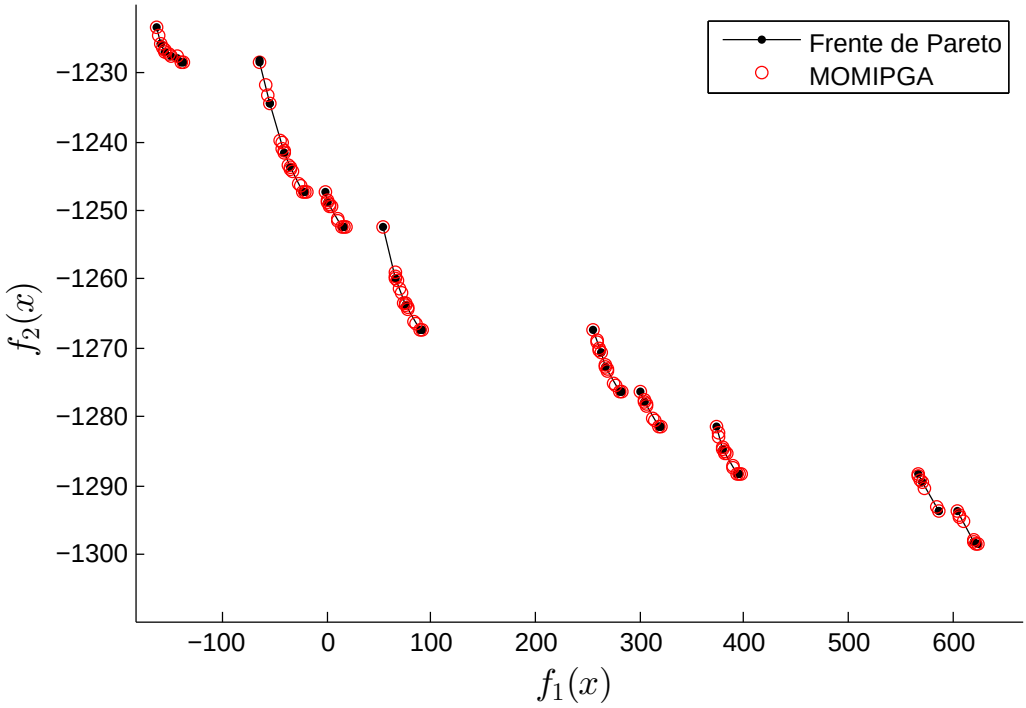

Figura 4.12: Frente de Pareto real y aproximado para el Problema C80-12

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	235	242	240.7 (2.43)	**241**	239
P. obtenidos	707	733	718.8 (4.47)	795	793
# llamadas a función	113,761	90,481	**103,593.16** (6,768.44)	1,835,948	613,880
Costo medio por punto	160.90	123.43	**144.11**	2,309.36	772.17
Tiempo	45.54	40.53	**43.07** (1.30)	90.58	135.43
Δ_2	0.0387	0.0045	0.0071 (0.0089)		

Tabla 4.13: Resultados de los métodos MOMIPGA y TSM para el Problema C80-12.

Problema 80-13

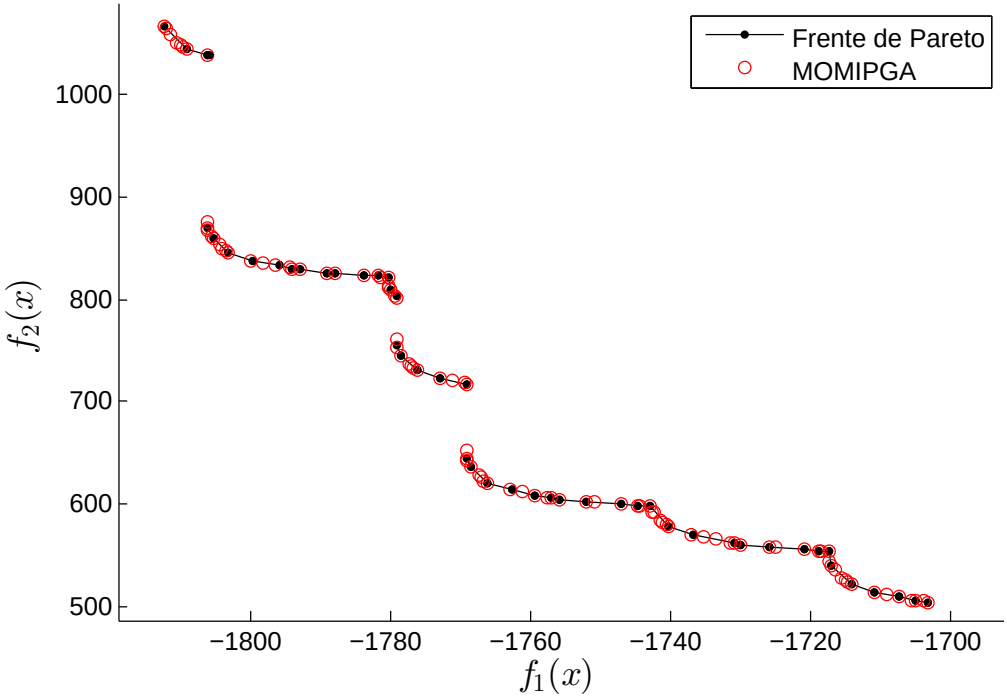

Figura 4.13: Frente de Pareto real y aproximado para el Problema C80-13

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	400	415	408.45 (3.81)	**411**	410
P. obtenidos	921	958	945.46 (9.01)	1223	1260
# llamadas a función	131,423	106,424	**119,337.93** (5,832.49)	5,307,692	1,142,143
Costo medio por punto	142.69	111.08	**126.22**	4,339.89	906.46
Tiempo	58.62	52.27	**53.90** (1.29)	406.20	277.03
Δ_2	0.0716	0.0172	0.0286 (0.0151)		

Tabla 4.14: Resultados de los métodos MOMIPGA y TSM para el Problema C80-13.

Problema 80-14

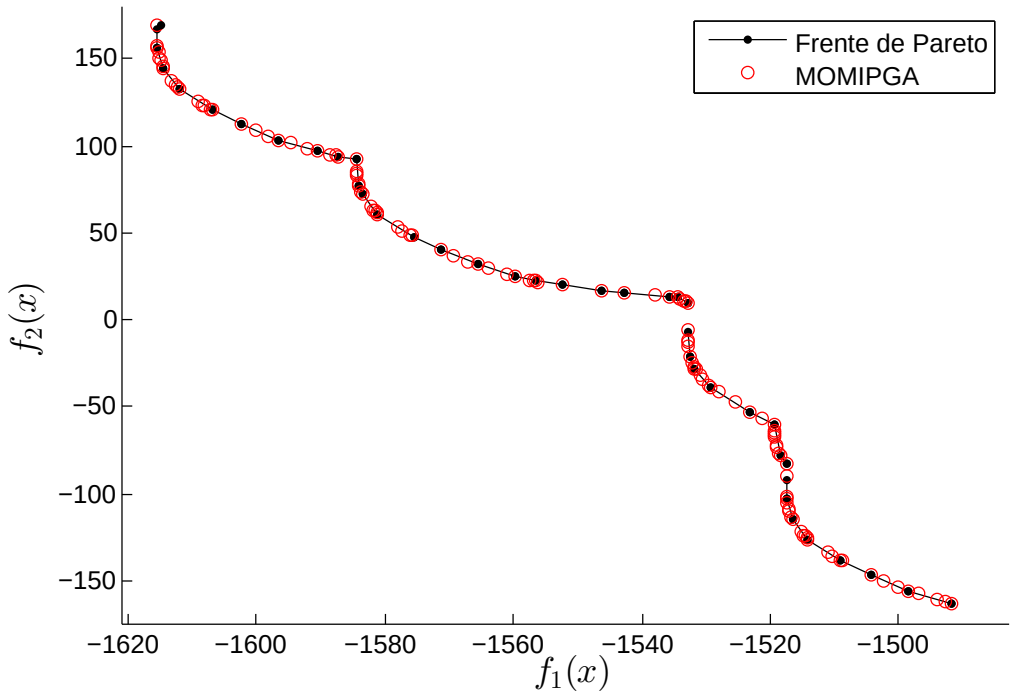

Figura 4.14: Frente de Pareto real y aproximado para el Problema C80-14

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	320	327	**325.67** (2.07)	322	322
P. obtenidos	951	961	956.56 (2.92)	1148	1157
# llamadas a función	89,218	70,435	**82,626.26** (4,771.67)	3,500,226	971,232
Costo medio por punto	93.81	73.29	**86.37**	3,048.97	839.43
Tiempo	49.56	42.79	**44.92** (1.59)	253.34	231.58
Δ_2	0.0233	0.0060	0.0117 (0.0054)		

Tabla 4.15: Resultados de los métodos MOMIPGA y TSM para el Problema C80-14.

Problema 80-15

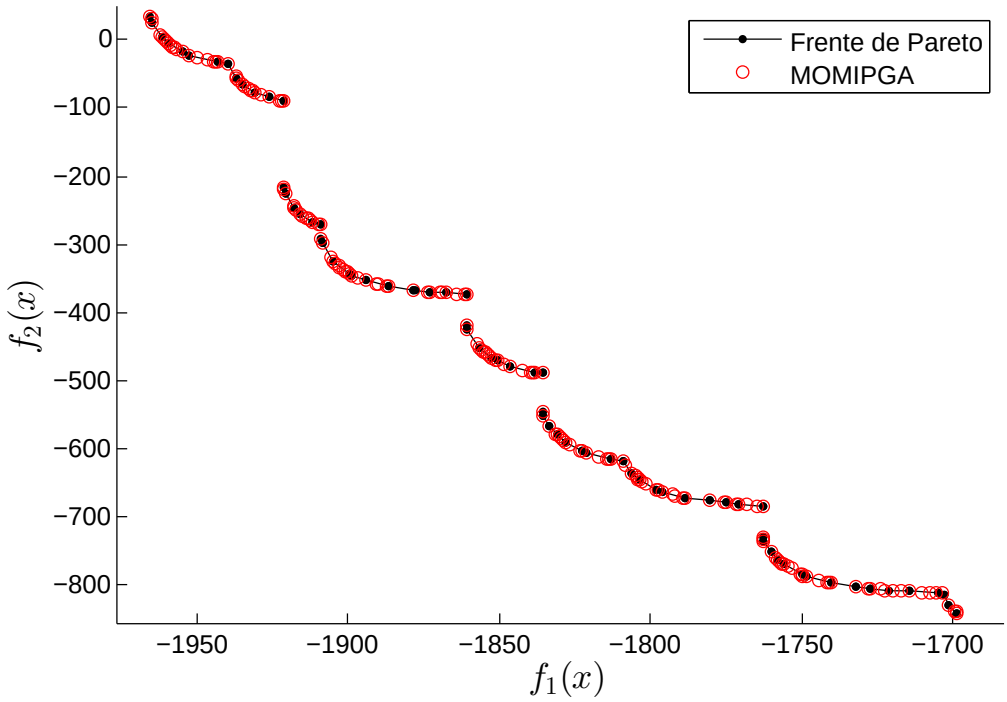

Figura 4.15: Frente de Pareto real y aproximado para el Problema C80-15

	MOMIPGA			TSM-Lp_solve	TSM-Gurobi
	Peor	Mejor	Promedio (D. estándar)		
P. extremos	240	247	245.89 (3.81)	**247**	**247**
P. obtenidos	760	768	764.63 (2,3116)	919	954
# llamadas a función	89,099	70,366	**78,502.26** (3,956.41)	4,043,379	684,823
Costo medio por punto	117.23	91.62	**102.66**	4,399.75	717.84
Tiempo	48.77	38.96	**40.58** (1.93)	276.09	158.46
Δ_2	0.0208	0.0061	0.0107 (0.0047)		

Tabla 4.16: Resultados de los métodos MOMIPGA y TSM para el Problema C80-15.

De las 15 instancias que se analizaron se puede ver por los resultados obtenidos que en 10 de ellas (i.e., C20-1, C20-2, C20-3, C20-4, C20-5, C40-7, C40-9, C40-10, C80-11, C80-14) el algoritmo propuesto MOMIPGA es capaz de generar en promedio una mejor solución (más puntos sobre el frente de Pareto) y con un costo computacional, en términos del número de evaluaciones de la función requeridas para obtener un punto del frente de Pareto, considerablemente más bajo en comparación con el método estado del arte. En las cinco instancias restantes (i.e., C40-6, C40-8, C80-12, C80-13, C80-15) MOMIPGA también generó una mejor solución al problema, en su mejor ejecución, y en promedio dio un resultado competitivo con respecto al método TSM con un número mucho menor de recursos en cuanto a evaluaciones de la función y con un tiempo comparable de ejecución. Esto se debe a que para el número de puntos extremos reportados en MOMIPGA se consideran sólo aquellos puntos que se obtiene con exactitud y no una aproximación a ellos. Por lo que a pesar de que en estas instancias el indicador Δ_p reporta un buen valor, los puntos extremos obtenidos en promedio son menores a los que se obtienen ejecutando el método TSM. De las instancias donde MOMIPGA es superior se puede destacar en particular la instancia C80-14, pues MOMIPGA determina en promedio 325 puntos sobre el frente de Pareto con únicamente el 8.5 % del número de evaluaciones de la función objetivo que tiene que realizar TSM con `Gurobi® Optimizer` para sólo obtener 322 puntos. Además, como es de esperarse en este caso, el tiempo de ejecución es menor para MOMIPGA.

Por otra parte, en términos del tiempo de ejecución se observa que mientras el número de variables aumenta, MOMIPGA se vuelve más eficiente. Esto debido al diseño que se siguió para desarrollar MOMIPGA, pues el gran beneficio poblacional que se hereda de los algoritmos genéticos es capaz de analizar distintas combinaciones sobre las variables discretas en una sola ejecución mientras que en esa misma ejecución TSM debe analizar variable por variable a través de una estructura de árbol[1].

[1]Recordar que TSM usa una paquetería para resolver problemas lineales mixtos y que éstas resuelven el problema usando BBMs.

Capítulo 5

Conclusiones y trabajo futuro

En este trabajo se propuso el algoritmo denominado MOMIPGA, el cuál es un método heurístico para tratar problemas de optimización lineal mixta multiobjetivo. La propuesta que aquí presentamos se enfoca en problemas con dos objetivos, sin embargo todas las partes del método, salvo el módulo de hibridación con el TSM, pueden utilizarse en problemas con más objetivos. La idea central fue usar el poder poblacional de los AGs para buscar eficientemente las combinaciones de valores en las variables discretas que contribuyen al frente de Pareto, en lugar de hacerlo de manera individual como tradicionalmente ocurre en los BBMs. En la etapa de pruebas el algoritmo propuesto mostró ser capaz de generar el frente de Pareto completo en un tiempo menor a los algoritmos basados en BBM. Además, en comparación con el TSM, el algoritmo resolvió el problema utilizando considerablemente menos recursos computacionales e incluso logró correctamente el resultado buscado incluso para aquellos problemas donde TSM presentó deficiencias.

Durante la etapa de diseño de MOMIPGA fue necesario adaptar los operadores tradicionales del AG para lograr la eficiencia, aprovechando las propiedades geométricas del problema. Otra novedad en este algoritmo es el uso del Método Multisimplex. Esto permitió al motor de búsqueda del AG tomar ventaja de la cercanía a componentes conectadas del frente de Pareto, así como el acercar ciertas soluciones a regiones prometedoras de búsqueda, logrando así un ahorro significativo en los cálculos.

Al tratarse de un método heurístico, se llevó a cabo una experimentación rigurosa sobre problemas de prueba internacionalmente aceptados. Se analizaron quince instancias con hasta ochenta restricciones sobre ochenta variables. En todos los casos el algoritmo MOMIPGA logró generar todos los puntos buscados en el frente de Pareto

de cada problema, de una manera eficiente. En diez de los casos la ventaja eficiente se mantuvo para el promedio sobre 30 ejecuciones del algoritmo. En los cinco casos restantes, se obtuvo en el promedio un resultado competitivo con un ahorro considerable de recursos, y el resultado óptimo se obtuvo solo en las mejores ejecuciones de esa instancia. En algunos casos, la reducción en términos del número de llamadas a las funciones objetivo se observó en más de un ochenta por ciento, con respecto al costo con el método TSM. Con estas pruebas se comprobó la robustez del algoritmo y su eficiencia en esta clase de problemas.

Durante el desarrollo de este trabajo se analizaron previamente 4 algoritmos que fueron explícitamente diseñados para resolver MOMIP's. Durante el análisis se encontró que los primeros dos algoritmos, cuyos autores son Mavrotas y Diakoulaki, tienen serias dificultades para encontrar el frente de Pareto e incluso están incompletos pues no son capaces de obtener todo el frente. En la investigación se encontró otro algoritmo que complementa las deficiencias de los 2 anteriores. A pesar de ello, los 3 están basados en técnicas de ramificación y acotamiento, lo cual por su naturaleza combinatoria los hace computacionalmente muy lentos, y por lo mismo se limitan a problemas con pocas variables. Al final se analizó el algoritmo denominado TSM cuyo objetivo es obtener el frente de Pareto completo. Sin embargo éste también presenta serias dificultades, pues para su uso es indispensable contar con una paquetería de optimización mixta, al cual el algoritmo es altamente sensible. Además este método es muy susceptible a errores de redondeo que hacen que el algoritmo pierda ciertos puntos importantes para la definición del frente de Pareto completo.

Como trabajo a futuro se plantea la posibilidad de adaptar el módulo de MOMIP-GA correspondiente al TSM para atacar problemas con tres objetivos. Otro enfoque es llevar problemas lineales mixtos con más de dos objetivos a planteamientos equivalentes en solo dos objetivos, siguiendo ciertas rutas preferenciales dadas por el usuario.

Bibliografía

[gur, 2015] (2015). Gurobi optimization, inc. Disponible en http://www.gurobi.com. Ultimo acceso 5 de junio de 2016.

[fic, 2016] (2016). Fico xpress optimization suite. Disponible en http://www.fico.com. Ultimo acceso 5 junio del 2016.

[Antunes et al., 2004] Antunes, C. H., Martins, A. G., and Brito, I. S. (2004). A multiple objective mixed integer linear programming model for power generation expansion planning. *Energy*, 29(4):613–627.

[Bateson, 1901] Bateson, W. (1901). Experiments in plant hybridization por Gregor Mendel. *Journal of the Royal Horticultural Society*, 24(1):1–32. Traducción al inglés del artículo de Gregor Mendel.

[Bazaraa et al., 2011] Bazaraa, M. S., Jarvis, J. J., and Sherali, H. D. (2011). *Linear programming and network flows*. John Wiley & Sons.

[Berkelaar et al., 2004] Berkelaar, M., Eikland, K., and Notebaert, P. (2004). lp_solve 5.5, open source (mixed-integer) linear programming system. Software. Disponible en http://lpsolve.sourceforge.net/5.5/. Ultimo acceso 5 junio de 2016.

[Boland et al., 2014] Boland, N., Charkhgard, H., and Savelsbergh, M. (2014). The triangle splitting method for biobjective mixed integer programming. In *Integer Programming and Combinatorial Optimization*, pages 162–173. Springer.

[Bui and Alam, 2008] Bui, L. T. and Alam, S. (2008). *Multi-objective optimization in computational intelligence: theory and practice*. Information Science Reference Hershey, PA.

[Coello and Cortés, 2005] Coello, C. A. C. and Cortés, N. C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190.

[Coello et al., 2002] Coello, C. A. C., Van Veldhuizen, D. A., and Lamont, G. B. (2002). *Evolutionary algorithms for solving multi-objective problems*, volume 242. Springer.

[Daniel, 2009] Daniel, B. (2009). Xpress-optimizer reference manual. *Fair Isaac Corporation, Leamington Spa, Warwickshire, UK*.

[Darwin, 1859] Darwin, C. (1859). *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*.

[Das and Dennis, 1997] Das, I. and Dennis, J. E. (1997). A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural optimization*, 14(1):63–69.

[De Zulueta, 2009] De Zulueta, A. (2009). *El origen de las especies por medio de la selección natural*. Editorial CSIC-CSIC Press. Traducción del trabajo de Charles Darwin al español.

[Deb, 2014] Deb, K. (2014). Multi-objective optimization. In Burke, E. K. and Kendall, G., editors, *Search Methodologies*, pages 403–449. Springer US.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197.

[Deep et al., 2009] Deep, K., Singh, K. P., Kansal, M., and Mohan, C. (2009). A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Applied Mathematics and Computation*, 212(2):505–518.

[Demirtas and Özden Üstün, 2008] Demirtas, E. A. and Özden Üstün (2008). An integrated multiobjective decision making process for supplier selection and order allocation. *Omega*, 36(1):76 – 90. Special Issue Section: Papers presented at the {INFORMS} conference, Atlanta, 2003.

[Donoso and Fabregat, 2010] Donoso, Y. and Fabregat, R. (2010). *Multi-objective optimization in computer networks using metaheuristics*. CRC Press.

[Egon Balas, 1965] Egon Balas, Fred Glover, S. Z. (1965). An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, 13(4):517–549.

[Ehrgott, 2005] Ehrgott, M. (2005). *Multicriteria optimization*. Springer Science & Business Media, 2nd edition.

[Eiben and Smith, 2010] Eiben, A. E. and Smith, J. E. (2010). *Introduction to evolutionary computing*, volume 2. Springer Berlin.

[Fogel et al., 1966] Fogel, L. J., Owens, A. J., and Walsh, M. J. (1966). *Artificial Intelligence through Simulated Evolution*. John Wiley.

[Garey and Johnson, 1990] Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

[Haimes et al., 1971] Haimes, Y. Y., Ladson, L., and Wismer, D. A. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-1(3):296–297.

[Hathhorn et al., 2013] Hathhorn, J., Sisikoglu, E., and Sir, M. Y. (2013). A multi-objective mixed-integer programming model for a multi-floor facility layout. *International Journal of Production Research*, 51(14):4223–4239.

[Holland, 1975] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[Howlett, 1989] Howlett, P. (1989). Arguments on evolution. a paleontologist's perspective. *Geobios*, 22(5):691.

[Koza, 1994] Koza, J. (1994). *Genetic Programming II*. MIT Press, Cambridge, MA.

[Koza, 1991] Koza, J. R. (1991). *Genetic Programming*. MIT Press.

[Luenberger and Ye, 2008] Luenberger, D. G. and Ye, Y. (2008). *Linear and nonlinear programming*. Springer, cuarta edition.

[Marler and Arora, 2004] Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395.

[Mavrotas and Diakoulaki, 1998] Mavrotas, G. and Diakoulaki, D. (1998). A branch and bound algorithm for mixed zero-one multiple objective linear programming. *European Journal of Operational Research*, 107(3):530–541.

[Mavrotas and Diakoulaki, 2005a] Mavrotas, G. and Diakoulaki, D. (2005a). A mixed integer multiple objective linear programming model for capacity expansion in an autonomous power generation system. In Loulou, R., Waaub, J.-P., and Zaccour, G., editors, *Energy and Environment*, pages 191–210. Springer US.

[Mavrotas and Diakoulaki, 2005b] Mavrotas, G. and Diakoulaki, D. (2005b). Multicriteria branch and bound: A vector maximization algorithm for mixed 0-1 multiple objective linear programming. *Applied Mathematics and Computation*, 171(1):53–71.

[Mendel, 1866] Mendel, G. (1866). Versuche über pflanzenhybriden. *Verhandlungen des naturforschenden Vereines in Brunn 4: 3*, 44.

[Miettinen, 1998] Miettinen, K. (1998). *Nonlinear Multiobjective Optimization*, chapter A Posteriori Methods, pages 77–113. Springer US, Boston, MA.

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. (2006). *Numerical optimization*. Springer Science & Business Media.

[Osyczka, 1984] Osyczka, A. (1984). *Multicriterion optimization in engineering with FORTRAN programs*. Ellis Horwood series in mechanical engineering. E. Horwood.

[Pareto, 1971] Pareto, V. (1971). Manual of political economy.

[Pasternak and Passy, 1973] Pasternak, H. and Passy, U. (1973). Bicriterion ma–thematical programs with boolean variables. En J. L. Cochrane and M. Zeleny, editors. *Multiple Criteria Decision Making*, pages 327–348.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag.

[Rothlauf, 2002] Rothlauf, F. (2002). *Representations for genetic and evolutionary algorithms*. Springer.

[Schütze et al., 2012] Schütze, O., Esquivel, X., Lara, A., and Coello, C. A. C. (2012). Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 16(4):504–522.

[Schwefel, 1995] Schwefel, H.-P. (1995). *Evolution and optimum seeking.* Sixth-generation computer technology series. Wiley.

[Singh and Kumar, 2012] Singh, A. and Kumar, S. (2012). Multiple objectives mathematical programming using payoff techniques. *International Journal Pure Applied Science and Technology*, 9(1):39–46.

[Stadler, 1988] Stadler, W. (1988). Fundamentals of multicriteria optimization. In *Multicriteria Optimization in Engineering and in the Sciences*, pages 1–25. Springer.

[Steuer, 1989] Steuer, R. E. (1989). *Multiple criteria optimization; theory, computation, and application*. Number 1 in Wiley Series in Probability and Mathematical Statistics-Applied. Wiley Online Library.

[Vincent et al., 2013] Vincent, T., Seipp, F., Ruzika, S., Przybylski, A., and Gandibleux, X. (2013). Multiple objective branch and bound for mixed 0-1 linear programming: Corrections and improvements for the biobjective case. *Computers & Operations Research*, 40(1):498–509.

[Vira and Haimes, 1983] Vira, C. and Haimes, Y. Y. (1983). *Multiobjective Decision Making: Theory and Methodology*. Elsevier Sciencie Publishing Co.

[Waltz, 1967] Waltz, F. M. (1967). An engineering approach: hierarchical optimization criteria. *Automatic Control, IEEE Transactions on*, 12(2):179–180.