UNIVERSIDAD DE ALMERIA

OPTIMIZACIÓN GLOBAL INTERVALAR: PREDICCIÓN, ACOTACIÓN Y SEPARABILIDAD

Tesis doctoral presentada por
JOSÉ LUIS BERENGUEL GÓMEZ

Dirigida por
Dr. LEOCADIO GONZÁLEZ CASADO
y
Dr. ELIGIUS M. T. HENDRIX

UNIVERSIDAD DE ALMERIA

OPTIMIZACIÓN GLOBAL INTERVALAR: PREDICCIÓN, ACOTACIÓN Y SEPARABILIDAD

Tesis doctoral presentada por
JOSÉ LUIS BERENGUEL GÓMEZ

Dirigida por
Dr. LEOCADIO GONZÁLEZ CASADO
y
Dr. ELIGIUS M. T. HENDRIX

El doctorando | El director | El director

Almería, octubre de 2012

Optimización global intervalar: predicción, acotación y separabilidad

Doctorando: José Luis Berenguel Gómez

Director: Dr. Leocadio González Casado

Director: Dr. Eligius M.T. Hendrix

La siguiente página web contiene información actualizada sobre los temas relacionados con esta tesis doctoral:

http://www.hpca.ual.es/

Texto impreso en Almería, octubre 2012

A mis abuelos

Prefacio

La vida cotidiana está llena de problemas de optimización: la organización de productos dentro de un almacén, la organización de los paquetes y productos en las furgonetas de una empresa de transportes, la fabricación de envases con las dimensiones óptimas para maximizar el almacenamiento y/o minimizar el material empleado, el diseño de piezas industriales que se ajusten a unas determinadas cualidades o el encontrar la mezcla adecuada de distintas materias primas de tal modo que se mejore el producto final y se minimice su coste son algunos ejemplos de problemas de optimización. La calidad de la solución hallada permitirá ahorrar en costes de producción y fabricación, mejorar el tiempo de fabricación, mejorar la organización y gestión de los recursos, obtener mejores productos con un menor coste, etc. Las áreas donde se hace necesaria la resolución de problemas de optimización son muy diversas: Ingeniería, Química, Economía, Robótica, etc.

Podemos distinguir dos tipos de problemas de optimización desde un punto de vista estrictamente matemático: la optimización lineal y la optimización no lineal. Los problemas de optimización lineal son aquellos en los que únicamente intervienen funciones lineales. Son bien conocidos y en general su dificultad depende del número de restricciones lineales. Los problemas de optimización no lineal pertenecen a la clase de problemas NP-Completos, es decir, no existe una solución en tiempo polinómico, por lo que a medida que el número de variables del problema aumenta, el tiempo para resolverlo crece exponencialmente. Una segunda distinción de los problemas de optimización se realiza entre optimización combinatoria y optimización continua, aunque ambas clases de problemas pueden aparecer mezclados. Esta tesis se enmarca

dentro de los problemas de optimización continua no lineal donde el valor de las variables varía libremente dentro de unos límites.

En general, un problema de optimización puede contener muchas soluciones que son válidas o *factibles* dentro de los parámetros o de la región de búsqueda, conocidas como óptimos locales. Los algoritmos que encuentran soluciones locales son denominados algoritmos de optimización local. La mejor solución de todas las locales es el óptimo global y la búsqueda del mismo se conoce como *optimización global*. Comúnmente, la búsqueda del óptimo global consiste en la búsqueda del mínimo valor global de la función que queremos minimizar, puesto que los problemas de maximización pueden reescribirse como uno de minimización. El mejor valor, u óptimo global, de la función en la región de búsqueda es único, pero puede encontrarse en distintas localizaciones, lo que hace que el problema sea aún más complicado. Esta tesis se enmarca en el ámbito de la optimización global.

Una clasificación clásica de las estrategias existentes para resolver los problemas de optimización global las divide en dos: estrategias de búsqueda estocásticas y estrategias de búsqueda deterministas. Las estrategias de búsqueda estocásticas se basan en muestrear la función objetivo con un conjunto finito de puntos iniciales escogidos al azar y en la generación de nuevos puntos aleatorios basándose en criterios heurísticos. Estos métodos no aseguran que el mínimo global sea obtenido al finalizar el proceso de búsqueda si el número de muestras no es lo suficientemente grande, o dicho de otra forma, solo garantizan el encontrar la solución global cuando el número de muestras tiende a infinito. Asimismo, este tipo de estrategias suelen utilizar un número elevado de iteraciones del algoritmo para tratar de mejorar la solución obtenida en iteraciones anteriores. Ese número de iteraciones suele establecerse al inicio. Las estrategias de búsqueda deterministas generan una secuencia de pasos fijos cuando el algoritmo se ejecuta para el mismo problema. Estas estrategias pueden obtener una aproximación del

óptimo global con la precisión deseada, aunque no pueden garantizar que su localización sea la correcta.

Los métodos que exploran todo el espacio de búsqueda se conocen como *métodos exhaustivos*. Estos métodos aseguran la localización del mínimo global con la precisión deseada al finalizar el proceso. Uno de los métodos exhaustivos más usados son los algoritmos de *ramificación y acotación* (*branch-and-bound*). Este algoritmo construye un árbol de búsqueda en donde los nodos activos corresponden a regiones del espacio de búsqueda donde puede encontrarse la solución global. El nodo más prometedor es el primero en visitarse y dividirse. Los criterios para elegir el nodo más prometedor se basan en cuestiones heurísticas y se conoce como estrategia de selección. Si en la estrategia de selección no influyen factores aleatorios, el algoritmo tendrá un comportamiento determinista. Si aquellos nodos que han sido rechazados o eliminados son aquellos en los que se ha verificado que no contienen una solución global, la solución obtenida será una solución *rigurosa*.

Utilizaremos la *aritmética de intervalos* para garantizar que los nodos eliminados no contienen el óptimo global. Con la aritmética de intervalos podemos obtener cotas superior e inferior del rango real de la función objetivo. Si un nodo tiene una cota inferior mayor que el mejor valor global del mínimo conocido hasta el momento, podemos descartar ese nodo ya que en su interior no encontraremos la solución. Además, otras muchas de las propiedades de la aritmética de intervalos permiten acelerar la resolución del problema, como por ejemplo el *test de monotonía*. Al final del proceso se obtiene una lista de nodos que determina la región o regiones donde se localiza el mínimo de la función y cotas superiores e inferiores de su valor. Cuanto mayor sea la precisión deseada para la solución, menor será el tamaño de la región o regiones obtenidas en donde se encuentra el mínimo global y mayor el tiempo de cómputo necesario para encontrarla.

Los algoritmos de ramificación y acotación intervalares son uno de los pocos métodos que obtienen una solución rigurosa a los problemas de optimización global. Esta tesis se enmarca en el estudio de estos métodos exhaustivos, rigurosos y deterministas.

Los algoritmos de ramificación y acotación pueden requerir una enorme cantidad de cómputo y memoria. Debido a estos grandes requerimientos, se deben utilizar mecanismos que aceleren el proceso de búsqueda y minimicen los requisitos de memoria. El proceso de mejora de los algoritmos de ramificación y acotación puede ir orientado a descubrir nuevas técnicas o procedimientos en el ámbito secuencial del algoritmo de optimización, o bien, a estudiar nuevos métodos y estrategias que presenten algoritmos paralelos que hagan uso de múltiples procesadores de forma eficiente.

En el ámbito secuencial, las mejoras pueden ir encaminadas a obtener mejores funciones de inclusión que obtengan intervalos más ceñidos al rango real de la función, a diseñar nuevas reglas de eliminación que permitan evaluar un número inferior de nodos o aprovechar las cualidades individuales de un problema de optimización concreto para diseñar nuevos métodos y algoritmos más eficientes. En general, cualquier aspecto del algoritmo de optimización global es susceptible de ser mejorado.

El reto principal al que uno se enfrenta cuando paraleliza un algoritmo de ramificación y acotación es la gestión adecuada de la carga de trabajo de los procesadores. El balanceo de la carga persigue maximizar el tiempo de cómputo útil de cada uno de los procesadores tratando de evitar que estos estén ociosos. La forma óptima de conseguir esto es determinar los recursos computacionales que son necesarios para cada problema en cada momento de la ejecución. En esta tesis estudiaremos algunos mecanismos para tratar de predecir el número de nodos que quedan por evaluar por el algoritmo de ramificación y acotación. Este valor se puede emplear para determinar los recursos computacionales necesarios en cada momento de la ejecución en este tipo de algoritmos.

El objetivo general de esta tesis es encontrar nuevos mecanismos que permitan acelerar la convergencia de los algoritmos de ramificación y acotación intervalares, tanto secuenciales como paralelos.

La organización de esta tesis es como sigue. El capítulo 1 introduce y clasifica los problemas de optimización y enmarca en esta clasificación a los problemas de optimización global continua, en los que estamos interesados. El capítulo 2 describe las características y propiedades fundamentales de la aritmética de intervalos. El capítulo 3 estudia en profundidad la resolución de problemas de optimización global mediante algoritmos de ramificación y acotación intervalares con una extensa revisión bibliográfica. En el capítulo 4 se estudia la estimación de la carga computacional en los algoritmos de ramificación y acotación. Este valor puede emplearse para decidir los recursos computacionales necesarios para distribuir la carga de trabajo entre varios procesadores, lo que puede mejorar el rendimiento de los algoritmos paralelos de ramificación y acotación. Se presentan tres estrategias diferentes y se estudia la calidad de las mismas. Los resultados obtenidos en las predicciones muestran que se pueden obtener buenas estimaciones del número de nodos pendientes por evaluar a partir de la mitad de la ejecución y en algunos casos, incluso desde etapas más tempranas. El capítulo 5 presenta un nuevo método de acotación basado en la separación de términos de la función objetivo. Se desarrolla toda la base teórica para funciones unidimensionales y se estudia su rendimiento sobre un conjunto de funciones de prueba. Los resultados obtenidos son bastante prometedores. Se pretende extender este estudio a funciones de más de una dimensión en trabajos futuros. El capítulo 6 presenta un nuevo algoritmo secuencial de ramificación y acotación para funciones objetivo aditivamente separables con variables comunes, se estudian sus propiedades y se presentan nuevos mecanismos de aceleración. Los resultados empíricos muestran que se puede mejorar la eficiencia en la optimización de una función aditivamente separable con variables comunes con este algoritmo específico.

Agradecimientos

Ha llegado el momento de los reconocimientos. Si hoy me encuentro escribiendo estas líneas es gracias a la ayuda y el apoyo de muchas personas. He aquí mi agradecimiento por la inestimable ayuda que he recibido.

Todo es gracias a ellos, las personas más importantes sin las cuales no habría sido posible: Dr. Leocadio González Casado, Dr. Eligius M.T. Hendrix, Dra. Inmaculada García Fernandez y Dr. Frédéric Messine.

Incluso estas líneas no son suficientes para mostrar todo el agradecimiento hacia Leo. Él me inició en el mundo de la investigación a través de una beca de colaboración del Ministerio de Educación y Ciencia. Continué la senda a su lado durante mi Proyecto Fin de Carrera donde me enseñó todo lo relacionado con la optimización global y los algoritmos de ramificación y acotación intervalares. Durante el Trabajo Fin de Máster fue cuando surgió la idea de tratar de estimar la carga de trabajo pendiente en un algoritmo de ramificación y acotación y la semilla para todo el trabajo y esfuerzo que vino después. Muchos años de trabajo juntos que por fin ha dado sus frutos.

Me encuentro sin palabras para describir la admiración y el aprecio que siento por Eligius e Inma. Desde el comienzo se volcaron conmigo y me ayudaran a sacar adelante todo el trabajo que se plasma en esta tesis. En los momentos de más dificultad, cuando el trabajo parece estar en punto muerto, es cuando más necesitado está uno de sabios consejos. Y es lo que siempre he recibido de vosotros.

A Frédéric Méssine lo conocí durante una de las conferencias del máster en Técnicas Informáticas Avanzadas en la Universidad de Almería, pero fue durante el EURO del año 2009 en Bonn cuando empezamos a

trabajar juntos en la idea de un algoritmo específico para funciones separables. Su colaboración, sus ideas y sus consejos han sido fundamentales. ¡Lo próximo será aprender a tocar la guitarra!

Recién terminado el Proyecto Fin de Carrera, me incorporé al grupo de investigación "TIC-146 Supercomputación: Algoritmos". Durante todos estos años sus integrantes han sido como mis compañeros de trabajo, aunque mi presencia a vuestro lado no haya sido demasiado continua. Siempre habéis estado ahí cuando lo he necesitado.

Indudablemente, todo el trabajo desempeñado para la consecución de esta tesis no hubiera sido posible sin la ayuda y financiación de los organismos públicos, en primer lugar mi agradecimiento al Ministerio de Ciencia e Innovación por la finaciación del proyecto TIN2008-01117 en el que participo, y en segundo lugar a la Junta de Andalucía y al Fondo Europeo de Desarrollo Regional (FEDER) por la financiación de los proyectos P08-TIC-3518 y P11-TIC-7176 en los que también estoy involucrado.

Si alguien debe tener un agradecimiento especial es mi familia: mis abuelos han sido todo un ejemplo en mi infancia y es el ejemplo en el que me gustaría verme cuando envejezca, mis padres que me han dado todo lo que ha estado en sus manos y más, y mis hermanos que hacen la vida más divertida. Y a ti Marisa, que eres mi apoyo en todo lo que hago y siempre estás a mi lado.

A los que formáis parte de mi vida y de mi día a día y a los que desgraciadamente ya no estáis a mi lado.

A todos, gracias.

José Luis Berenguel Gómez

9 de octubre de 2012

Contenidos

Índice de figuras

Índice de tablas

Todo comienzo tiene su encanto.
Johann Wolfgang von Goethe

CAPÍTULO
1
Introducción

1.1 Introducción a la optimización

La optimización matemática, también conocida como programación matemática, busca la respuesta a la pregunta *"¿qué es mejor?"* en problemas donde la calidad de la respuesta se puede expresar mediante un valor.

Un ejemplo clásico de este tipo de problemas es el problema del viajante de comercio (*Travelling Salesperson Problem*)[3, 4]. El enunciado del problema es el siguiente:

> Dadas N ciudades de un territorio, el objetivo es encontrar una ruta que, comenzando y terminando en una ciudad concreta, pase una única vez por cada ciudad y minimice la distancia recorrida por el viajante.

A pesar de la aparente sencillez del enunciado, este problema es NP-Completo, es decir, su complejidad crece de forma no polinomial con la dimensión, en este caso, con el número de ciudades a visitar. Por ejemplo, para solo 12 ciudades existen $479\,001\,600$ rutas posibles. Este es un ejemplo típico de optimización combinatoria.

Otro ejemplo, en este caso de optimización continua, puede ser el problema de encerrar un conjunto de puntos con una forma geométrica determinada de modo

que el volumen de la misma sea mínimo. La forma geométrica puede ser un hiperrectángulo [71] o una esfera [72].

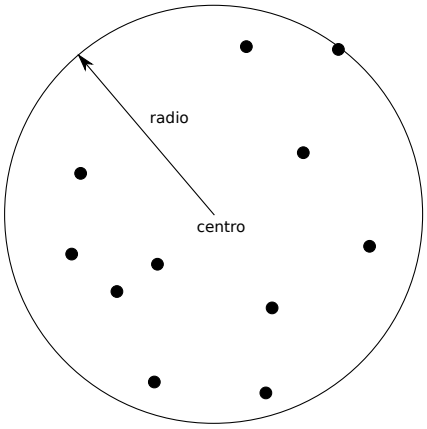

Figura 1.1: Esfera de mínimo volumen - Conjunto de puntos encerrado por una esfera de mínimo volumen.

Problemas de optimización como los mostrados anteriormente se encuentran y tienen su aplicación práctica en campos tan diversos como la Ingeniería Aeronáutica, la Ingeniería de Materiales, la Ingeniería Industrial, Economía, Matemática, Física, Biología, Química, etc.

1.2 Clasificación de los problemas de optimización

Podemos distinguir entre dos tipos de problemas de optimización:

- Problemas de caja blanca. Son aquellos en los que la expresión analítica de la función objetivo está disponible.

- Problemas de caja negra. Son aquellos en los que no está disponible la expresión analítica de la función objetivo. Para obtener un valor de la función, hay que pasar la lista de parámetros a una función o rutina que devuelve el valor de la función tras un cierto tiempo. Este tipo de problemas también aparecen en diseños industriales de forma que, solo una vez desarrollado el producto a partir de una serie de parámetros, se puede obtener la calidad del resultado. Por ejemplo, en el diseño de los escapes de un motor solo se puede saber la

potencia alcanzada después de chequear el motor con el nuevo escape en un banco de pruebas.

Esta distinción es importante ya que dependiendo del tipo de problema podemos, o no, analizar la estructura del mismo y tomar decisiones en base a ello, tratando de utilizar el algoritmo que mejor comportamiento muestre para cada estructura. En esta tesis trataremos únicamente los problemas de caja blanca.

Una clasificación general, dependiente del número de variables, de su tipo y del tipo de funciones involucradas en el problema de optimización se describe en la tabla 1.1.

Tabla 1.1: Clasificación de los problemas de optimización

Característica	**Propiedad**	**Clasificación**
Número de variables	Una	Unidimensional
	Varias	Multidimensional
Tipo de variables	Reales continuas	Continua
	Enteras	Entera o discreta
	Reales continuas y enteras	Entera mezclada
	Permutaciones de enteros	Combinatoria
Tipo de Funciones	Funciones lineales	Lineal
	Funciones cuadráticas	Cuadrática
	Otras funciones no lineales	No Lineal
Formulación del problema	Sujeto a restricciones	Con restricciones
	No sujeto a restricciones	Sin restricciones

Por ejemplo, cuando las variables de los problemas de optimización son variables enteras, estos problemas se llaman también problemas de programación entera (IP, *Integer Programming*). Si las funciones del problema son lineales se llaman problemas de programación lineal (LP, *Linear Programming*). Si por el contrario, dichas funciones no son lineales, nos encontramos ante problemas de programación no lineal (NLP, *Non Linear Programming* [7]), etc. En esta tesis estudiamos mejoras para los algoritmos de optimización empleados en resolver problemas de programación no lineal con variables continuas y sin restricciones.

Una enumeración más concreta puede encontrarse en [44, 61, 115], donde además existen numerosas referencias de los siguientes problemas de optimización: programación bilineal o biconvexa, optimización combinatoria, minimización cóncava, optimización global continua, diferencial convexa, programación fraccional, problemas lineales y no lineales complementarios, optimización *Lipschitz*, problemas minimax, optimización multinivel, programación multiobjetivo, programación multiplicativa, problemas de redes de trabajos, programación paramétrica no convexa, optimización cuadrática, programación inversa convexa, optimización global separable y otros modelos probabilísticos no convexos. Para resolver estos problemas se utilizan modelos o algoritmos de optimización que tratan de aprovechar las características y estructuras propias y singulares de cada tipo de problema.

En cuanto a la complejidad de la resolución de estos problemas, Pardalos y Schnitger [111] mostraron que demostrar que un posible punto candidato es un óptimo local, para problemas de programación cuadrática, es un problema NP-Completo. Horst, Pardalos y Thoai [62] demostraron que el problema de complementariedad lineal es un problema NP-Completo. Ahmadi et al. [1] estudiaron la complejidad de determinar si un polinomio de grado 4 con n variables describe una función convexa [113]. Demostraron que para polinomios de grado par mayores o iguales que cuatro, este problema es NP-Completo. Pardalos y Vavasis [112], demostraron también que la programación cuadrática con un autovalor negativo es un problema NP-Completo. Resolver el problema no convexo más simple de programación no lineal, es NP-Completo. Esto indica, que el tiempo de resolución de cualquier algoritmo, en el peor caso, se incrementa exponencialmente con el tamaño del problema (teoría acerca de NP-Completitud puede encontrarse en [45]).

1.3 Clasificación de los algoritmos de optimización

Una clasificación clásica de las estrategias existentes para resolver los problemas de optimización global las divide en dos: estrategias de búsqueda estocásticas y estrategias de búsqueda deterministas. Las estrategias de búsqueda estocásticas se basan en muestrear la función objetivo con un conjunto finito de puntos iniciales escogidos al azar y en la generación de nuevos puntos aleatorios basándose en criterios heurísticos. Estos métodos no aseguran que el mínimo global sea obtenido

al finalizar el proceso de búsqueda si el número de muestras no es lo suficientemente grande, o dicho de otra forma, solo garantizan el encontrar la solución global cuando el número de muestras tiende a infinito [127]. Asimismo, este tipo de estrategias suelen utilizar un número elevado de iteraciones del algoritmo para tratar de mejorar la solución obtenida en iteraciones anteriores. Ese número de iteraciones suele establecerse al inicio. Las estrategias de búsqueda deterministas indexbúsqueda!determinista pueden encontrar la solución con una precisión deseada pero pueden no garantizar la localización del mismo, como por ejemplo el método DIRECT [57].

Otro tipo de clasificación más actual realizada por Neumaier [106] se basa en la rigurosidad de la solución obtenida por el algoritmo de optimización. Se diferencia entre:

Métodos incompletos. Aquellos que pueden quedar atrapados en un mínimo local.

Métodos asintóticamente completos. Aquellos que alcanzan siempre el mínimo global en una ejecución suficientemente larga, pero que no tienen información acerca de cuando un mínimo global es encontrado.

Métodos completos. Aquellos que alcanzan con certeza el mínimo global (en ausencia de errores en los cálculos) en una ejecución suficientemente larga y que además conocen una aproximación a la solución global durante su ejecución.

Métodos rigurosos. Aquellos que alcanzan con certeza el mínimo global para la precisión deseada incluso cuando existen errores en los cálculos.

Los algoritmos de ramificación y acotación intervalares, que se estudiarán en esta tesis, pertenecen a la categoría de métodos rigurosos.

1.4 El problema de la optimización global

El objetivo en un problema de optimización global es encontrar la mejor solución de forma que se minimice o maximice el valor de una función objetivo, posiblemente sujeta a una serie de restricciones sobre los posibles candidatos. Como el

máximo global de una función f es el mínimo global de $-f$, los problemas de maximización o minimización son equivalentes, por lo que consideraremos únicamente el problema de minimización. Además, estamos interesados en obtener todos los puntos para los que la función tiene el valor mínimo.

La figura 1.2 ilustra un problema de optimización global con múltiples mínimos locales.

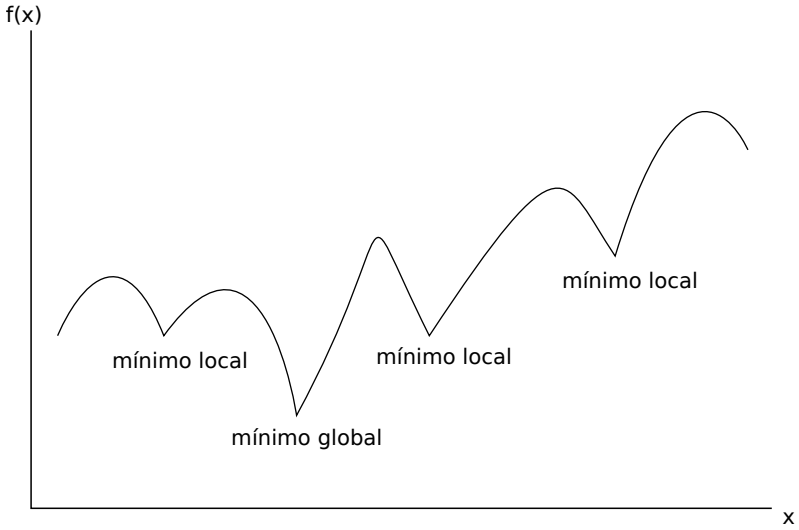

Figura 1.2: Problema de optimización global con múltiples mínimos locales - La función $f(x)$ contiene varios mínimos locales pero un único mínimo global.

La descripción general de este problema es:

$$\begin{aligned} minimizar \quad & f(x) \\ sujeto\, a: \quad & g(x) \leq 0, \\ & h(x) = 0, \\ & x \in S. \end{aligned} \tag{1.1}$$

donde $f(x) : \mathbb{R}^n \to \mathbb{R}$ es la *función objetivo*, $g(x) : \mathbb{R}^n \to \mathbb{R}^m$ y $h(x) : \mathbb{R}^n \to \mathbb{R}^k$ son funciones que restringen la región de búsqueda y S el dominio del problema. Una solución global del problema (1.1) viene dada por $x^* \in S$ de forma que $f^* = f(x^*) \leq f(x),\ \forall x \in S$.

Cuando $g(x)$ y $h(x)$ no existen, el problema de optimización se suele incluir en el grupo de problemas denominados sin restricciones. Aunque se denominan

así en algunos trabajos, realmente la búsqueda se restringe a un espacio de búsqueda (*unconstrained global optimization* o *box-constrained global optimization*). Por lo tanto, el problema que queremos resolver en esta tesis es encontrar el grupo de puntos x^* tal que:

$$f(x^*) = \min_{x \in S} f(x) \tag{1.2}$$

donde $f : S \subset \mathbb{R}^n \rightarrow \mathbb{R}$ es una función continua, aunque el trabajo presentado aquí puede extenderse para funciones no continuas. Asumiremos que la región de búsqueda S es compacta, es decir, cerrada y acotada sin más restricciones que las definidas por los límites del espacio de busqueda ($x \in S$). Denotaremos por f^* al mínimo global de la función ($f^* = f(x^*)$).

Claro que lo entiendo. Incluso un niño de cuatro años podría entenderlo. ¡Que me traigan un niño de cuatro años!

Groucho Marx

CAPÍTULO

2

Aritmética de intervalos

2.1 Introducción

La aritmética de intervalos apareció como una forma de evitar los errores de redondeo producidos por el computador. Estos errores se deben a la representación y aritmética de números en punto flotante definida en el estándar IEEE-754 en 1985 y revisado en 2008 [66]. En la actualidad, el grupo de trabajo P1788 del IEEE prepara la estandarización de la aritmética de intervalos computacional[1].

La aritmética de intervalos es una generalización de la aritmética real donde los números, representados por intervalos, reemplazan a los números reales y la aritmética de intervalos reemplaza a la aritmética real.

Los resultados teóricos obtenidos mediante el análisis numérico están basados en números exactos y en una aritmética exacta, mientras que en casos prácticos aparece la necesidad del redondeo cuando los números no son representables en la computadora. El propósito inicial de la aritmética de intervalos era obtener unos límites superior e inferior de las soluciones teniendo en cuenta los errores cometidos en la aritmética computacional. Varios autores trabajaron sobre la idea de limitar los errores de la aritmética computacional mediante intervalos [38, 129].

[1] http://grouper.ieee.org/groups/1788/

La aritmética de intervalos se atribuye a Ramon E. Moore en su tesis doctoral [96] y posteriormente con la aparición de su libro *Interval analysis* en 1966 [97]. Este libro ha sido revisado recientemente en 2009 [102]. Se pueden encontrar las referencias a los trabajos previos que realizó Moore sobre la aritmética de intervalos en la web de R. B. Kearfott[1]. Moore transformó esta simple idea en una herramienta para el análisis del error. En vez de tratar solo errores de redondeo, Moore extendió el uso de la aritmética de intervalos para limitar el efecto de los errores de todas clases, incluyendo errores de aproximación y errores en los datos. A partir del trabajo de Moore han aparecido numerosos artículos y libros dedicados a este tema [2, 53].

2.2 Ejemplo de errores de redondeo.

El fallo de un misil Patriot en la guerra del Golfo de 1991 y la explosión del cohete Arian 5 varios segundos después de su lanzamiento en 1996, son desastres supuestamente debidos a errores de la aritmética computacional.[2] Para enfatizar más sobre el problema, a continuación se mostrará un ejemplo debido a Rump [126]. La siguiente ecuación:

$$f(x,y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y} \qquad (2.1)$$

fue evaluada para $x = 77\,617$ e $y = 33\,096$. El programa para su cálculo fue escrito en Fortran. Este fue compilado y ejecutado sobre una SparcStation SLC. La exactitud de las tres precisiones probadas, simple, doble y extendida fue de 6, 14 y 35 dígitos decimales, respectivamente. Solo se usaron las operaciones básicas, por lo que las potencias se realizaron mediante multiplicaciones. Los resultados obtenidos fueron los siguientes:

$$\begin{aligned} \text{(precisión simple) f} &= 6.33825 \cdot 10^{29} \\ \text{(precisión doble) f} &= 1.1726039400532 \\ \text{(precisión extendida) f} &= 1.17260394005317863185883490452018 38 \end{aligned}$$

[1] http://interval.louisiana.edu/Moores_early_papers/

[2] http://www.ima.umn.edu/∼arnold/disasters/

Observando los resultados, uno puede concluir que el resultado en simple precisión es erróneo. Además, uno puede (erróneamente) concluir que el resultado en doble precisión es preciso, ya que concuerda en 13 dígitos con el resultado en precisión extendida. Sorprendentemente para muchos, todos los resultados son incorrectos, incluso en el primer dígito ya que el signo es incorrecto. En [83] se habla también de este ejemplo. El resultado exacto fue obtenido mediante la aritmética de intervalos implementada en el software VPI [40] usando una precisión variable con 40 dígitos de exactitud. El resultado está incluido en el intervalo:

$$[-0.827396059946821368114116509547981629200 5,$$
$$-0.827396059946821368114116509547981629198 6].$$

Si el resultado se hubiera calculado usando aritmética de intervalos con poca precisión, se hubiera obtenido un intervalo menos preciso, pero que contendría el resultado correcto. Este ejemplo fue revisado en [85] usando la aritmética IEEE-754 y el entorno Forte Developer 6 update 2 para Fortran 95 de Sun Microsystem Inc. y los resultados obtenidos fueron incluso peores.

La aritmética de intervalos puede usarse para calcular límites del rango real de una función como se ilustra en la figura 2.1. El intervalo obtenido incluye el rango real para la función f en un intervalo X (véase la sección 2.5 para más información sobre funciones de inclusión). La cota o límite superior está representada por $\overline{F}(X)$ y el límite inferior por $\underline{F}(X)$. F es la extensión a intervalos de f y $\overline{F}(m(X))$ es una cota superior de f evaluada en el punto medio de X. Se puede observar cómo el valor del rango real de la función $f(x)$ en el intervalo X ($f(X)$) está entre los límites inferior y superior obtenidos por $F(X)$.

Una área del análisis de intervalos donde todavía se está trabajando bastante es el desarrollo de algoritmos de intervalos que produzcan resultados que se ciñan lo más posible al rango real de una función en un intervalo [86, 87, 123].

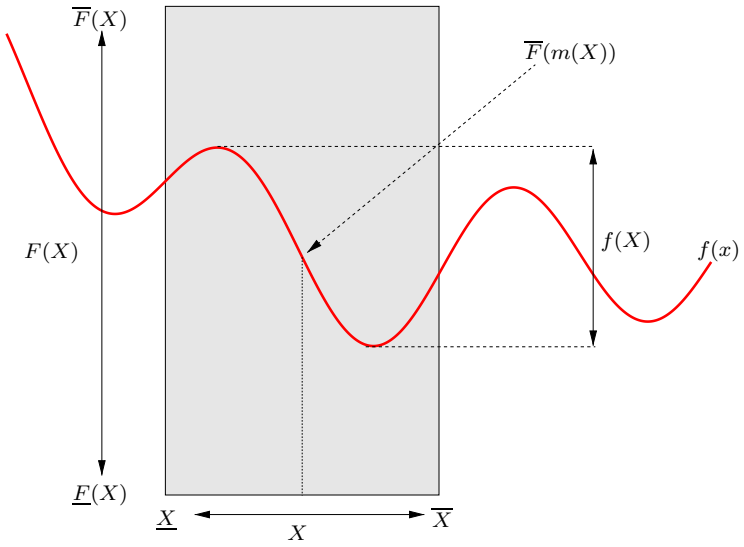

Figura 2.1: Función de inclusión - Obtención de un intervalo que incluye el rango real de la función f en X, mediante aritmética de intervalos.

2.3 Definiciones y operaciones básicas de la aritmética de intervalos

Definición 1 *Un intervalo cerrado denotado por* $[a, b]$ *es el conjunto de los números reales dados por*

$$[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}.$$

Por convención, denotaremos a los intervalos con letras mayúsculas y al conjunto de intervalos con $\mathbb{I}$. Los extremos izquierdo y derecho de un intervalo $X \in \mathbb{I}$ se denotarán con $\underline{X}$ y $\overline{X}$ respectivamente.

Definición 2 *Se dice que un intervalo* X *es degenerado si* $\underline{X} = \overline{X}$.

Ejemplo 1 *El intervalo* $X = [3, 3]$ *es un intervalo degenerado.*

2.3.1 Intersección, unión y envolvente convexa

Definición 3 *Sean* X *e* Y *dos intervalos, se define la intersección* $X \cap Y$ *como el intervalo*

$$\begin{aligned} X \cap Y &= \{z : z \in X \wedge z \in Y\} \\ &= [\text{máx}\{\underline{X}, \underline{Y}\}, \text{mín}\{\overline{X}, \overline{Y}\}]. \end{aligned} \tag{2.2}$$

En caso de que los intervalos X e Y no tengan ningún punto z en común, esto es, si $\overline{Y} < \underline{X}$ o $\overline{X} > \underline{Y}$, el intervalo resultante de la intersección será el conjunto vacío $\emptyset$.

Definición 4 *Sean X e Y dos intervalos, se define la unión $X \cup Y$ como el intervalo*

$$\begin{aligned} X \cup Y &= \{z : z \in X \wedge z \in Y\} \\ &= [\text{mín}\{\underline{X}, \underline{Y}\}, \text{máx}\{\overline{X}, \overline{Y}\}], \end{aligned} \tag{2.3}$$

cuando existe un punto z en común en X e Y. En general la unión de dos intervalos no siempre es un único intervalo. Para evitar esto se define la *envolvente convexa* (o *interval hull*) de dos intervalos:

Definición 5 *Sean X e Y dos intervalos, se define la envolvente convexa $X \underline{\cup} Y$ como el intervalo*

$$X \underline{\cup} Y = [\text{mín}\{\underline{X}, \underline{Y}\}, \text{máx}\{\overline{X}, \overline{Y}\}]. \tag{2.4}$$

La envolvente convexa es siempre un intervalo y puede usarse en operaciones intervalares. Tenemos que

$$X \cup Y \subseteq X \underline{\cup} Y. \tag{2.5}$$

Ejemplo 2 *Sean los intervalos $X = [-3, 5]$ e $Y = [7, 12]$, entonces $X \underline{\cup} Y = [-3, 12]$. En cambio, $X \cup Y$ no puede representarse como un único intervalo ya que es un conjunto no conexo.*

2.3.2 Relaciones de orden entre intervalos

Así como en el conjunto de los números reales disponemos de la relación "menor que" ($<$), podemos definir esta misma operación sobre intervalos

$$X < Y \text{ si y solo si } \overline{X} < \underline{Y}. \tag{2.6}$$

Otra operación de orden muy útil en aritmética de intervalos es la inclusión de intervalos:

$$X \subseteq Y \text{ si y solo si } \underline{Y} \leq \underline{X} \text{ y } \overline{X} \leq \overline{Y}. \tag{2.7}$$

Ejemplo 3 *Tenemos que* $[-1, 5] \subseteq [-5, 5]$. *Esta operación no siempre se cumple, dado que si los intervalos se solapan, no podemos establecer ninguna relación de orden, por ejemplo para* $X = [0, 6]$ *e* $Y = [1, 10]$.

2.3.3 Operaciones aritméticas

Las operaciones básicas de suma, resta, multiplicación y división sobre intervalos se calculan de la siguiente manera.

$$X + Y = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}]. \tag{2.8}$$

La resta de dos intervalos se puede ver como la suma $X + (-Y)$ donde $-Y = [-\overline{Y}, -\underline{Y}]$, por tanto:

$$X - Y = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}]. \tag{2.9}$$

La multiplicación de dos intervalos viene dada por:

$$X \cdot Y = [\text{mín } S, \text{máx } S], \text{ donde } S = \{\underline{X} \cdot \underline{Y}, \underline{X} \cdot \overline{Y}, \overline{X} \cdot \underline{Y}, \overline{X} \cdot \overline{Y}\}. \tag{2.10}$$

La división de intervalos es un caso particular de la multiplicación donde:

$$X/Y = X \cdot \frac{1}{Y} = [\underline{X}, \overline{X}] \cdot [\frac{1}{\overline{Y}}, \frac{1}{\underline{Y}}], \text{ donde } 0 \notin Y. \tag{2.11}$$

La división de dos intervalos X/Y, donde $0 \in Y$, es posible en la aritmética de intervalos extendida, también llamada Kahan-Novoa-Ratz en honor a sus autores [67, 108, 121]. A continuación se muestra cómo se calcula la división en esta aritmética cuando $0 \in Y$:

$$X/Y = \begin{cases} [-\infty, +\infty] & \text{si } 0 \in X, \\ [\overline{X}/\underline{Y}, +\infty] & \text{si } \overline{X} < 0 \wedge \overline{Y} = 0, \\ [-\infty, \overline{X}/\overline{Y}] \cup [\overline{X}/\underline{Y}, +\infty] & \text{si } \overline{X} < 0, \\ [-\infty, \overline{X}/\overline{Y}] & \text{si } \overline{X} < 0 \wedge \underline{Y} = 0, \\ [-\infty, \underline{X}/\underline{Y}] & \text{si } 0 < \underline{X} \wedge \overline{Y} = 0, \\ [-\infty, \underline{X}/\underline{Y}] \cup [\underline{X}/\overline{Y}, +\infty] & \text{si } 0 < \underline{X} < 0, \\ [\underline{X}/\overline{Y}, +\infty] & \text{si } \underline{X} < 0 \wedge \underline{Y} = 0. \end{cases} \tag{2.12}$$

El cuadrado de un intervalo puede expresarse así:

$$X^2 = \begin{cases} [\underline{X}^2, \overline{X}^2], & 0 \leq \underline{X} \leq \overline{X}, \\ [\overline{X}^2, \underline{X}^2], & \underline{X} \leq \overline{X} \leq 0, \\ [0, \text{máx}\{\underline{X}^2, \overline{X}^2\}], & \underline{X} < 0 < \overline{X}. \end{cases} \tag{2.13}$$

El cuadrado de un intervalo, X^2, no es lo mismo que $X \cdot X$. Por ejemplo,

$$[-2, 2]^2 = [0, 4] \text{ mientras que } [-2, 2] \cdot [-2, 2] = [-4, 4].$$

En general, esta sobrestimación se produce al evaluar una expresión donde hay varias ocurrencias de una misma variable. Este fenómeno se conoce como la *dependencia de intervalos*.

2.3.4 Anchura, valor absoluto y punto medio

Definimos la anchura de un intervalo X y la denotamos por

$$w(X) = \overline{X} - \underline{X}. \tag{2.14}$$

Definimos el valor absoluto de X y lo denotamos por $|X|$ como el máximo de los valores absolutos de sus extremos:

$$|X| = \text{máx}\{|\underline{X}|, |\overline{X}|\}. \tag{2.15}$$

Por último, el punto medio de un intervalo X viene dado por

$$m(X) = \frac{\underline{X} + \overline{X}}{2}. \tag{2.16}$$

2.3.5 Vectores de intervalos

Un vector de intervalos de n dimensiones es una n-tupla ordenada de intervalos:

$$X = (X_1, \ldots, X_n) \in \mathbb{I}^n.$$

También lo denotaremos con letras mayúsculas, el subíndice indicará la componente del vector y nos referiremos a él habitualmente como "n-caja" o simplemente caja.

Todas las operaciones mostradas para intervalos se pueden extender con pequeñas modificaciones para vectores de intervalos. Por ejemplo:

1. Si $x = (x_1, \ldots, x_n)$ es un vector de reales y $X = (X_1, \ldots, X_n)$ es un vector de intervalos entonces

$$x \in X \text{ si } x_i \in X_i \; \forall i = 1, \ldots, n.$$

2. La intersección de dos vectores de intervalos es vacía si la intersección de alguno de sus componentes es vacía, es decir $X_i \cap Y_i = \emptyset$ para algún i. En otro caso tendríamos

$$X \cap Y = (X_1 \cap Y_1, \ldots, X_n \cap Y_n).$$

3. La relación de inclusión entre vectores de intervalos se define como

$$X \subseteq Y \text{ si } X_i \subseteq Y_i \; \forall i = 1, \ldots, n.$$

4. La anchura de un vector de intervalos es la de su componente más ancha, es decir,

$$w(X) = \text{máx}\{w(X_i)\} \; \forall i = 1, \ldots, n.$$

5. El punto medio de un vector de intervalos es el resultante de calcular los puntos medios de cada una de sus componentes:

$$m(X) = (m(X_1), \ldots, m(X_n)).$$

2.4 Propiedades de la aritmética de intervalos

La aritmética de intervalos mantiene intactas algunas propiedades de la aritmética de números reales, así la suma y la multiplicación de intervalos son conmutativas y asociativas, y el intervalo $0 = [0, 0]$ y $1 = [1, 1]$ son el elemento neutro de la suma y la multiplicación, respectivamente.

Por otro lado, debemos hacer notar que $-X$ no es el elemento inverso de la suma para X, de hecho el intervalo resultado en la operación $X - X$ es mucho más ancho que el original, salvo que $\underline{X} = \overline{X}$, es decir, que X sea un intervalo degenerado, donde el resultado será $[0, 0]$. Para el resto de casos el resultado será:

$$X - X = w(X) \cdot [-1, 1] \tag{2.17}$$

Ejemplo 4 *Sea el intervalo* $X = [3, 7]$*, entonces* $X - X = [-4, 4]$.

De igual modo $X/X = 1$ solo si $w(X) = 0$. En general,

$$X/X = \begin{cases} [\underline{X}/\overline{X}, \overline{X}/\underline{X}], \text{ si } 0 < \underline{X}, \\ [\overline{X}/\underline{X}, \underline{X}/\overline{X}], \text{ si } \overline{X} < 0. \end{cases} \tag{2.18}$$

La ley distributiva de la suma respecto al producto no se mantiene en la aritmética de intervalos, sin embargo existe una ley subdistributiva:

$$X \cdot (Y + Z) \subseteq X \cdot Y + X \cdot Z. \tag{2.19}$$

La ley distributiva solo funciona en casos especiales. En particular, para cualquier número real x, se cumple:

$$x \cdot (Y + Z) = x \cdot Y + x \cdot Z \tag{2.20}$$

La ley distributiva también se cumple si los intervalos sobre los que se distribuye la suma tienen el mismo signo:

$$X \cdot (Y + Z) = X \cdot Y + X \cdot Z \text{ si se cumple que } Y \cdot Z > 0 \tag{2.21}$$

Se dice que la aritmética de intervalos tiene la propiedad de inclusión isótona, esto es, sea $\odot \in \{+, -, \cdot, /\}$ y A, B, C y D intervalos tales que

$$A \subseteq C \text{ y } B \subseteq D,$$

entonces

$$A \odot B \subseteq C \odot D.$$

2.5 Funciones de inclusión sobre intervalos

Sea $f : \mathbb{R}^n \to \mathbb{R}$, nos interesa conocer el rango real de la función f sobre un intervalo X, esto es:

$$f(X) = \{f(x) : x \in X\}. \tag{2.22}$$

Para algunas funciones (2.22) es fácil de calcular. Por ejemplo, en funciones monótonas como $\exp(x)$ o $\log(x)$ basta calcular el valor de la función en los extremos del intervalo sobre el que queremos obtener el rango de la función, así tenemos que $\exp(X) = [\exp(\underline{X}), \exp(\overline{X})]$ y $\log(X) = [\log(\underline{X}), \log(\overline{X})]$ para $\underline{X} \geq 0$.

Consideremos ahora la siguiente función:

$$f(x) = 1 - x, x \in \mathbb{R}. \tag{2.23}$$

Podemos aplicar las operaciones de intervalos sobre la ecuación (2.23) obteniendo la siguiente función sobre intervalos:

$$F(X) = 1 - X, X = [\underline{X}, \overline{X}]. \tag{2.24}$$

Para la función (2.23), además se cumple que el rango real coincide con el rango de la extensión a intervalos de f en (2.24), es decir, $f(X) = F(X)$. En general, esto no es cierto siempre, y la sobrestimación del intervalo obtenido por la función de inclusión dependerá de como se haya escrito la función, debido a la ausencia de la ley distributiva y del elemento inverso. Veamos un ejemplo:

Ejemplo 5 *La función $f(x) = x(1-x)^2$ puede ser escrita también como $g(x) = x - x^2$. Ambas funciones son idénticas matemáticamente en la aritmética real, es decir, producen el mismo resultado. Por ejemplo, en el intervalo $x \in [0, 1]$, el rango real de las funciones es*

$$f([0, 1]) = g([0, 1]) = [0, \frac{1}{4}].$$

Ahora bien, si cambiamos las funciones f y g por su extensión natural a intervalos $F(X) = X \cdot (1-X)$ y $G(X) = X - X^2$ respectivamente, y calculamos su inclusión en el mismo intervalo, obtendremos $F([0, 1]) = [0, 1]$ y $G([0, 1]) = [-1, 1]$.

Para obtener intervalos lo más ajustados posibles a la solución real debemos aplicar técnicas que reduzcan la aparición de la misma variable lo máximo posible [27, 53]. El ejemplo 5 ilustra este concepto.

2.5.1 Propiedades de las funciones de inclusión sobre intervalos

A continuación mencionamos algunas propiedades y definiciones relevantes en la aritmética de intervalos.

Definición 6 *Sea $f(X)$ el rango de la función f en el intervalo X. Una función de intervalos $F : \mathbb{I}^n \to \mathbb{I}$ es una* función de inclusión *si y solo si $f(X) \subseteq F(X) = [\underline{F}(X), \overline{F}(X)]$.*

Definición 7 *Se dice que F es una función de inclusión isótona si para cualquier intervalo $X \subseteq Y$, se tiene que $F(X) \subseteq F(Y)$.*

La aritmética de intervalos permite la obtención de una función de inclusión isótona:

Propiedad 1 *Teorema fundamental del análisis de intervalos. Dada una función real $f : \mathbb{R}^n \to \mathbb{R}$, una extensión natural a intervalos de f, denotada por $F_{NIE} : \mathbb{I}^n \to \mathbb{I}$, donde los operandos reales se cambian por intervalos y las operaciones reales por sus correspondientes operaciones sobre intervalos, es una función de inclusión isótona [97].*

En [6, 55, 92, 131] se explican otras funciones de inclusión isótonas.

Para medir la calidad de una función de inclusión F de f se utiliza el concepto de *exceso de anchura* o *sobrestimaciones en los límites* (*excess-width*), calculado como:

$$w(F(X)) - w(f(X)).$$

Este concepto fue introducido por Moore [97]. A partir de él surge otro concepto muy importante: la *convergencia* de las funciones de inclusión [69].

Definición 8 *Una función de inclusión F de f se dice que es α-convergente, con $\alpha > 0$, si*

$$w(F(X)) - w(f(X)) = O(w(X)^\alpha),$$

de tal forma que existe una constante $c \geq 0$ tal que

$$w(F(X)) - w(f(X)) \leq c \cdot w(X)^\alpha, \ \forall X \in \mathbb{I}^n, \ c \in \mathbb{R}. \tag{2.25}$$

Es importante escoger funciones de inclusión con órdenes de convergencia tan altos como sea posible para acelerar los métodos de optimización. Cuando α es 1 o 2 decimos que la inclusión es de primer orden o de segundo orden, respectivamente. Detalles sobre la convergencia de las funciones de inclusión pueden encontrarse en [75, 117, 132].

A continuación presentamos dos propiedades de interés en problemas de optimización:

Propiedad 2 *Refinamiento de F sobre X. Si $\bigcup_{k=1}^{d} X^k$ es una división de X entonces $F(X) \supseteq \bigcup_{k=1}^{d} F(X^k)$, con respecto a la propiedad 1.*

Propiedad 3 *Refinamiento de $\underline{F}$ sobre X. Si $\bigcup_{k=1}^{d} X^k$ es una división de X entonces $\min_k\{\underline{F}(X^k)\} \geq \underline{F}(X)$ es un límite inferior válido de f sobre X. Esto es consecuencia directa de la propiedad 2 y es de interés en problemas de minimización.*

En la sección 3.3.2 se explicarán los métodos de división de intervalos más usados.

Propiedad 4 *Monotonía. Sea $(F'_{NIE})_i$ una función de inclusión de la derivada parcial de $f(x)$ respecto de x_i. Si $\exists i \;\; 0 \notin (F'_{NIE})_i(X)$, entonces la función no contienen ningún punto estacionario en X.*

La propiedad 4 es útil para comprobar si una función diferenciable puede contener un óptimo en el interior del intervalo X. Si se cumple, la función no tiene ningún punto estacionario en X por lo que el intervalo puede eliminarse si el extremo de X con menor valor no coincide con el extremo de la región de búsqueda.

2.5.2 Formas centradas

Otra opción para obtener funciones de inclusión de intervalos, a parte de la extensión natural a intervalos, son las llamadas *formas centradas* (*centered forms*) introducidas por Moore (véase [75] y [53]). Las formas centradas son interesantes ya que presentan un orden de convergencia cuadrático, mientras que la extensión natural a intervalos tiene un orden de convergencia lineal. Aunque hay más funciones de inclusión que utilizan minorantes lineales para acotar la función ([55, 92, 105, 123, 134]), aquí explicaremos las más relevantes para esta tesis: la forma del valor medio (*mean value form*), la forma de Baumann y la forma LBVF (*Linear Boundary Value Form* [105]).

2.5.2.1 Forma del valor medio

Definición 9 *Sea* $f : \mathbb{R}^n \to \mathbb{R}$ *una función diferenciable y* $F' : \mathbb{I}^n \to \mathbb{I}^n$ *una función de inclusión del gradiente* f' *de* f*. Se define* $T_1 : \mathbb{I}^n \to \mathbb{I}$ *como*

$$T_1(c, X) = f(c) + (X - c)^T \cdot F'(X), \tag{2.26}$$

donde $c \in X$*, se llama función de la forma del valor medio o simplemente forma del valor medio.*

Suele escogerse como punto c el punto medio del intervalo, $m(X)$. Baumann [6] demostró que puede escogerse el punto c de forma óptima, de modo que se maximice el límite inferior o se minimice el límite superior obtenido por $T_1(c, X)$. El cálculo de F' suele hacerse vía diferenciación automática [16].

2.5.2.2 Forma de Baumann

Baumann [6] demostró que los límites obtenidos por la ecuación (2.26) se podían mejorar encontrando el valor óptimo de c. Baumann probó que escoger $c = b^-$ obtiene el mejor límite inferior, donde:

$$b_i^- = \begin{cases} \dfrac{\underline{X_i}\overline{F'}(X_i) - \overline{X_i}\underline{F'}(X_i)}{\overline{F'}(X_i) - \underline{F'}(X_i)} & , 0 \in F'(X_i) \\ \overline{X_i} & , \overline{F'}(X_i) \leq 0 \\ \underline{X_i} & , \underline{F'}(X_i) \geq 0 \end{cases}$$

Así,

$$f(X) \geq \underline{T_1}(b^-, X). \tag{2.27}$$

Para obtener el mejor límite superior, debemos reflejar el punto b^- sobre el punto medio $m(X)$ obteniendo:

$$b^+ = m(X) + (m(X) - b^-).$$

La figura 2.2 ilustra las diferencias obtenidas en el cálculo de los límites inferior y superior, si escogemos el punto medio del intervalo (izquierda) o los puntos de Baumann (derecha) para un problema unidimensional. Estos puntos también pueden usarse para reducir las dimensiones del intervalo X en problemas de optimización global, como se estudia en [131].

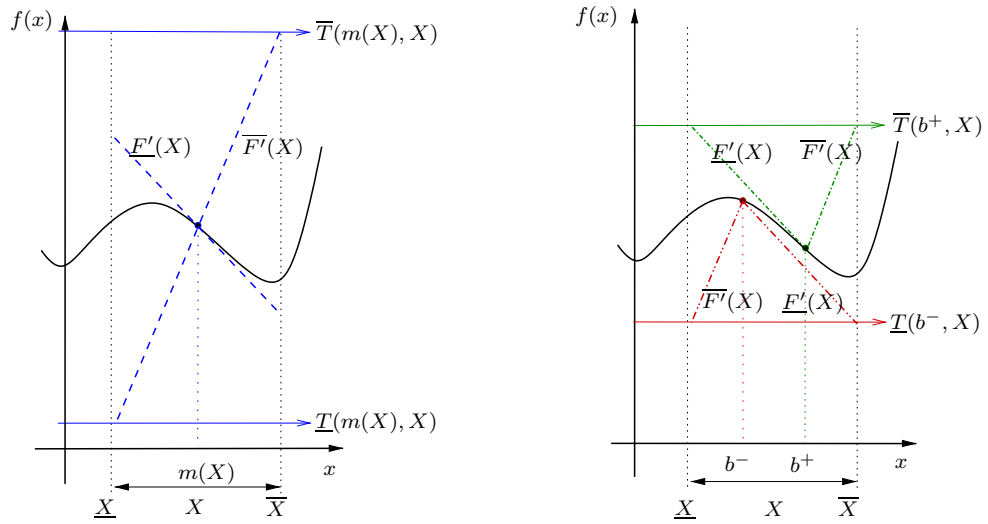

Figura 2.2: Forma centrada y forma de Baumann - Escoger los puntos de Baumann en lugar del punto medio, permite obtener una inclusión óptima de la forma centrada.

2.5.2.3 Forma Linear Boundary Value Form (LBVF)

Otra forma de obtener minorantes lineales basados en la derivada de la función objetivo es la llamada *Linear Boundary Value Form* (LBVF) que usa la evaluación de los puntos extremos del intervalo ([105] página 60 y [24, 56]).

Considerando el punto más a la izquierda de X, la función

$$\varphi l(x) = \underline{F}(\underline{X}) + \underline{F'}(X)(x - \underline{X}), \tag{2.28}$$

provee un minorante afín. De forma similar, el punto más a la derecha de X también provee un minorante afín

$$\varphi r(x) = \underline{F}(\overline{X}) - \overline{F'}(X)(\overline{X} - x) = \underline{F}(\overline{X}) + \overline{F'}(X)(x - \overline{X}). \tag{2.29}$$

Los valores $\underline{\varphi l}(\overline{X})$ y $\underline{\varphi r}(\underline{X})$ son límites inferiores de $f(X)$ en X. Se pueden obtener mejores límites inferiores cuando $0 \in F'(X)$ combinando las ecuaciones (2.28) y (2.29) en la función

$$\varphi m(x) = \text{máx}\{\varphi l(x), \varphi r(x)\}. \tag{2.30}$$

El límite inferior de la forma LBVF, $\underline{\varphi m}(X)$, se obtiene de encontrar el punto y en el que (2.28) y (2.29) son iguales:

$$y = \frac{\underline{F}(\underline{X}) - \underline{F}(\overline{X})}{w(F'(X))} + \frac{\overline{X} \cdot \overline{F'}(X) - \underline{X} \cdot \underline{F'}(X)}{w(F'(X))}. \tag{2.31}$$

La evaluación de (2.30) en y provee

$$\underline{\varphi m}(X) = \varphi m(y) = \frac{\underline{F}(\underline{X})\overline{F'}(X) - \underline{F}(\overline{X})\underline{F'}(X)}{w(F'(X))} + \frac{w(X)\overline{F'}(X)\underline{F'}(X)}{w(F'(X))}. \quad (2.32)$$

De manera similar, pueden construirse mayorantes lineales de $f(X)$. Usando el punto más a la izquierda de X se obtiene

$$\xi l(x) = \overline{F}(\underline{X}) + \overline{F'}(X)(x - \underline{X}), \quad (2.33)$$

y el punto más a la derecha X

$$\xi r(x) = \overline{F}(\overline{X}) - \underline{F'}(X)(\overline{X} - x) = \overline{F}(\overline{X}) + \underline{F'}(X)(x - \overline{X}). \quad (2.34)$$

Los valores $\overline{\xi l}(\overline{X})$ y $\overline{\xi r}(\underline{X})$ son límites superiores de $f(X)$. Un mejor mayorante puede obtenerse combinando ambas funciones.

$$\xi m(x) = \text{mín}\{\xi l(x), \xi r(x)\}. \quad (2.35)$$

El límite superior de la forma LBVF, $\overline{\xi m}(X)$, se obtiene de encontrar el punto z en el que las ecuaciones (2.33) y (2.34) son iguales:

$$z = \frac{\overline{F}(\overline{X}) - \overline{F}(\underline{X})}{w(F'(X))} + \frac{\underline{X}\overline{F'}(X) - \overline{X}\underline{F'}(X)}{w(F'(X))}, \quad (2.36)$$

y sustituyendo z en (2.35):

$$\overline{\xi m}(X) = \xi m(z) = \frac{\overline{F}(\overline{X})\overline{F'}(X) - \overline{F}(\underline{X})\underline{F'}(X)}{w(F'(X))} - \frac{w(X)\overline{F'}(X)\underline{F'}(X)}{w(F'(X))}. \quad (2.37)$$

Resumiendo:

$$f(X) \geq \underline{\varphi m}(X),\ 0 \in F'(X), \quad (2.38)$$

$$f(X) \leq \overline{\xi m}(X),\ 0 \in F'(X). \quad (2.39)$$

Las funciones de acotación $\varphi m(x)$ y $\xi m(x)$ se ilustran en la figura 2.3. Las formas de Baumann y LBVF ($\varphi m(x)$) pueden combinarse para obtener mejores límites inferiores como se estudió en [134].

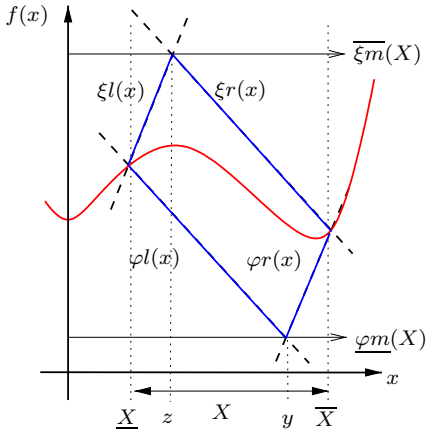

Figura 2.3: Forma Linear Boundary Value Form (LBVF) - Inclusión obtenida por la forma LBVF.

2.6 Software de aritmética de intervalos

Los experimentos y algoritmos de esta tesis se han implementado con el lenguaje C++ y la librería de aritmética de intervalos C-XSC (véase [50, 60]), que está disponible para su descarga en la web de la Universidad de Wuppertal[1].

Existen otras librerías de intervalos disponibles para este y otros lenguajes de programación como INTLAB, PROFIL/BIAS, FILIB++, MPFI y para paquetes de software matemático como MATLAB, Mathematica o Maple.

Una lista bastante extensa de librerías de intervalos y software relacionado con la computación verificada se puede ver en la web de la Universidad de Texas[2].

[1] http://www2.math.uni-wuppertal.de/∼xsc/

[2] http://www.cs.utep.edu/interval-comp/intsoft.html

Para mejorar nuestro conocimiento debemos aprender menos y contemplar más.

René Descartes

CAPÍTULO

3

Algoritmos de optimización global intervalares

3.1 Introducción a los métodos de ramificación y acotación

Little, Murty, Sweeny y Karel fueron los primeros que dieron el nombre de ramificación y acotación (*branch-and-bound*) a este tipo de algoritmos en su artículo innovador sobre el ya comentado problema del viajante de comercio en 1963 [84]. El método fue propuesto inicialmente por A. H. Land y A. G. Doig en 1960 para problemas de programación lineal entera (*ILP: Integer Linear Programming*) [80]. Lawler y Wood [82] estudiaron los algoritmos de ramificación y acotación, obteniendo una descripción independiente del problema, siendo así el primer artículo que presenta un método general. Los algoritmos de ramificación y acotación se han usado para resolver problemas NP-Completos ([39, 48, 80, 125]).

Los algoritmos de ramificación y acotación que hacen uso de la aritmética de intervalos son de los pocos métodos que garantizan la obtención del mínimo global y de los puntos minimizadores para el problema (1.1). Como resultado obtienen un intervalo que incluye el valor mínimo de la función y un conjunto de cajas que

contienen todos los puntos minimizadores. En [77] se describe una amplia variedad de clases de problemas para los que se pueden aplicar estos algoritmos.

El método de ramificación y acotación construye un árbol cuyos nodos o subproblemas son seleccionados, divididos, evaluados y almacenados o eliminados de acuerdo con una serie de reglas. El algoritmo consiste en una búsqueda heurística iterativa en el árbol que evita visitar subproblemas que no contienen una solución óptima. El algoritmo mantiene una cota superior del valor mínimo de la función que permite rechazar aquellos nodos que presentan un valor de la función objetivo por encima, es decir, no pueden contener una solución óptima. El número de nodos que se tienen que evaluar depende del problema a resolver y es desconocido a priori. Los algoritmos de backtracking, programación dinámica y las búsquedas en árboles A^* y *AND-OR* pueden verse como variaciones de algoritmos de ramificación y acotación [49, 78, 103, 104].

Según Ibaraki y Mitten [63, 95], los algoritmos de ramificación y acotación pueden ser caracterizados por las siguientes reglas:

- *División*: Divide un nodo de tal forma que este esté cubierto por la unión de los nodos resultantes.
- *Acotación*: Calcula cotas (superior y/o inferior) del valor del rango de la función objetivo para cada nodo.
- *Selección*: Determina cuál es el siguiente nodo a ser dividido.
- *Eliminación*: Establece cuándo un nodo no contiene una solución óptima y es eliminado del árbol de búsqueda.
- *Finalización*: Determina cuándo un nodo es solución, es decir, pertenece al conjunto final.

El algoritmo 1 describe un algoritmo general de ramificación y acotación. En primer lugar se inicializa la lista de trabajo Λ a la región de búsqueda y la lista de nodos solución Ω al conjunto vacío. La regla de selección escoge un nodo v de Λ y actualiza la cota superior del mínimo global $\overline{f^*}$ de la función f. A continuación, se divide v de acuerdo con la regla de división, generando los subnodos v^i. Se evalúa cada uno de estos subnodos y se comprueba si se pueden rechazar o si cumplen el criterio de finalización, en cuyo caso se almacenan en Ω; en caso contrario se vuelven a almacenar en Λ.

Algoritmo 1 : Algoritmo general de ramificación y acotación.
Funct Algoritmo_BB(S, f)

1. Inicializar la lista de trabajo $\Lambda := \{S\}$
2. Inicializar la lista final $\Omega := \emptyset$
3. Inicializar la cota superior del mínimo $\overline{f^*} = +\infty$
4. **mientras** ($\Lambda \neq \emptyset$)
5. Seleccionar un nodo v de Λ *Regla de selección*
6. Actualizar $\overline{f^*}$ si $\overline{F}(x) < \overline{f^*},\ x \in v$.
7. Dividir v generando v^i subnodos con $i = 1 \ldots k$ *Regla de división*
8. **desde** $i = 1$ hasta k
9. Evaluar v^i *Regla de acotación*
10. **si** v^i *no puede ser eliminado* *Regla de eliminación*
11. **si** v^i *satisface el criterio de finalización* *Regla de finalización*
12. Almacenar v^i en Ω
13. **si no**
14. Almacenar v^i en Λ
15. **devolver** Ω

3.2 Algoritmo clásico de ramificación y acotación intervalar

El algoritmo 1 muestra la estructura general de un algoritmo de ramificación y acotación. Este algoritmo se puede adaptar para resolver problemas de optimización global. Distinguiremos dos tipos de algoritmos de optimización global para resolver el problema (1.2), aquellos capaces de determinar (a) f^* o, (b) f^* y x^*. El algoritmo de Moore-Skelboe [128] es el primer algoritmo de ramificación y acotación intervalar capaz de determinar f^* (pero no x^*). Fue mejorado por Moore en [98]. El segundo algoritmo se debe a Ichida-Fujii [65] y trata de determinar f^* y x^*, pero este algoritmo tiene el problema de que la convergencia a x^* no está garantizada. Por último, el algoritmo presentado por Hansen [51, 52] obtiene una intervalo donde está incluido f^* y el conjunto de puntos minimizadores x^*. En [118] se describen estos algoritmos y se comparan numéricamente. El algoritmo 2, que se presenta a

Algoritmo 2 : Algoritmo intervalar de optimización global.
Funct Algoritmo_IBB(F, S, ϵ)

1. Inicializar $\Lambda := \{S\}$ y $\Omega := \emptyset$
2. Inicializar $\overline{f^*} = \overline{F}(m(S))$
3. **mientras** ($\Lambda \neq \emptyset$)
4. Seleccionar $X \in \Lambda$ con mejor $\underline{F}(X)$ *Regla de selección*
5. Calcular $\overline{F}(m(X))$
6. **si** $\overline{F}(m(X)) < \overline{f^*}$
7. $\overline{f^*} = \overline{F}(m(X))$
8. Borrar todos los $X \in \Lambda \cup \Omega$ con $\underline{F}(X) > \overline{f^*}$ *Test de corte*
9. Dividir X en subintervalos $X^j,\ j = 1, \ldots, k$ *Regla de división*
10. **desde** $j = 1$ hasta k
11. Calcular un límite inferior $\underline{F}(X^j)$ de $f(X^j)$ *Regla de acotación*
12. **si** $\underline{F}(X^j) > \overline{f^*}$ *Regla de eliminación*
13. Eliminar X^j
14. **si no si** $w(X^j) \leq \epsilon$ *Regla de finalización*
15. Insertar X^j en Ω
16. **si no**
17. Insertar X^j en Λ
18. **devolver** Ω

continuación, se basa en el algoritmo de Hansen.

Para resolver el problema (1.2), el algoritmo 2 utiliza la extensión a intervalos de la función objetivo F (véase la sección 2.5, página 17), el espacio o dominio de búsqueda S y la precisión de los nodos solución basados en su anchura, ϵ. Λ es el árbol de búsqueda donde se almacenan los nodos activos y Ω es la lista de nodos finales que cumplen la regla de terminación.

El algoritmo comienza inicializando la lista de trabajo Λ, la lista final Ω y la cota superior del mínimo $\overline{f^*}$ (línea 2). A continuación comienza una búsqueda iterativa en Λ (línea 3) que consiste en seleccionar el nodo X con el menor valor de su límite inferior $\underline{F}(X)$ (línea 4) y tratar de actualizar el mejor valor conocido de la cota superior del mínimo $\overline{f^*}$ evaluando el punto medio de la caja (línea 5).

Si $\overline{f^*}$ es actualizado (línea 7), eliminaremos las cajas de Λ y Ω cuyo $\underline{F}(X) > \overline{f^*}$; esto se conoce como test de corte (línea 8). El siguiente paso consiste en dividir el nodo seleccionado en varios subnodos, el criterio más utilizado es dividir X por su coordenada más ancha, generando dos nuevos nodos. A este procedimiento se le conoce con el nombre de bisección (línea 9). A cada uno de los nuevos nodos le calculamos una cota inferior del rango real mediante la función de inclusión F que utiliza aritmética de intervalos (línea 11) y se comprueba si el nodo puede ser rechazado (línea 12). Se pueden aplicar test adicionales que pueden permitir eliminar un mayor número de nodos (véase la sección 3.3.4) con un coste superior en el tiempo de cómputo. Si el nodo no ha sido rechazado, se inserta en la lista de nodos finales si su anchura ha alcanzado la precisión deseada ϵ (línea 15) o en la lista de trabajo (línea 17). Una vez finalizado el algoritmo, el conjunto de puntos mínimos x^* está encerrado en $\cup_{X\in\Omega} X$, y el óptimo global f^* está incluido en el intervalo $[\text{mín}_{X\in\Omega}\, \underline{F}(X), \overline{f^*}]$. En la sección 3.3 se presentarán los estudios más relevantes realizados sobre las distintas reglas de los algoritmos de ramificación y acotación intervalares para mejorar la eficiencia y eficacia del algoritmo 2.

La figura 3.1 muestra un árbol de ramificación y acotación generado por el algoritmo 2. La imagen está dividida por una línea punteada. Todos los nodos que están por encima de esa línea representan el estado actual del árbol, mientras que los nodos por debajo de la línea representan cómo quedará el árbol tras la finalización del algoritmo. Los nodos en color azul son los nodos almacenados en el árbol de búsqueda Λ en este momento de la ejecución. Los nodos de color gris son aquellos que han sido o serán rechazados, los nodos blancos son aquellos que serán o han sido divididos y los nodos hoja, de color rojo, son los nodos que pueden contener el conjunto de puntos óptimos x^*.

3.3 Estudios relevantes del algoritmo clásico

El algoritmo 2 ha sido y es ampliamente estudiado en la literatura. Las investigaciones se centran en mejorar su rendimiento a partir de modificaciones de las diferentes reglas del mismo. A continuación, describiremos los métodos y técnicas más relevantes.

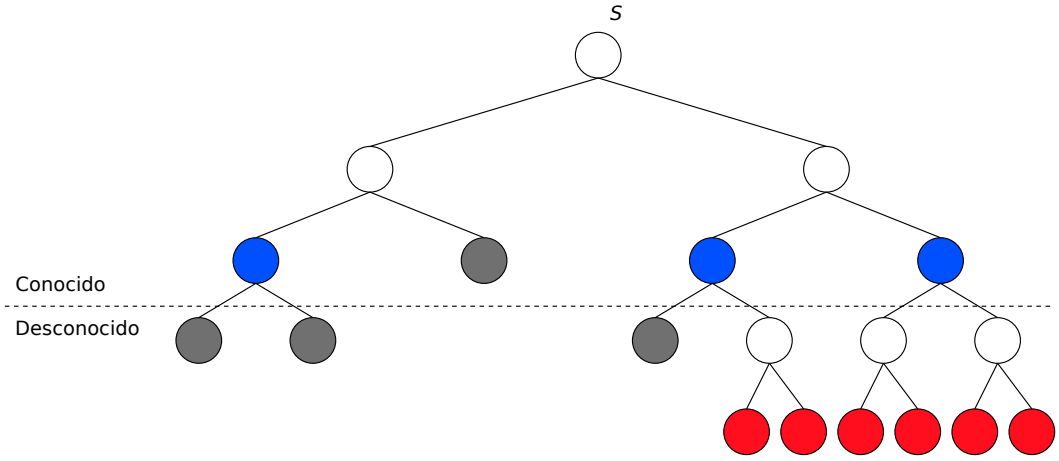

Figura 3.1: Árbol de ramificación y acotación - Árbol generado por un algoritmo de ramificación y acotación.

3.3.1 Estudios sobre la regla de selección

Una de las cuestiones fundamentales para que el algoritmo 2 sea eficiente es dedicar la mayor parte del trabajo sobre los nodos más prometedores. Las estrategias más comunes son aquellas propias de la búsqueda en árboles:

- Estrategia *primero el más antiguo* (*oldest-first*). Se selecciona el nodo más antiguo de Λ, es decir, Λ se comporta como una cola FIFO (*First In, First Out*).

- Estrategia *primero el más nuevo* o en profundidad (*depth-first*). Se selecciona uno de los nodos recientemente creados, es decir, Λ se comporta como una pila o cola LIFO (*Last In, First Out*).

- Estrategia *primero el mejor* (*best-first*). Se selecciona el nodo X con el menor límite inferior $\underline{F}(X)$, es decir, Λ se comporta como una lista con prioridad.

La estrategia *primero el más antiguo* tiene como ventaja la simplicidad del manejo de la estructura de datos pero su principal desventaja es que no se seleccionan los nodos más prometedores. La estrategia *primero el más nuevo* es similar a la anterior, el manejo de la lista es muy sencillo, además, tiende a mantener pocos nodos en la misma. Como principal desventaja, tiende a dividir y evaluar más nodos de

los necesarios, lo que también puede influir negativamente en el rendimiento. La estrategia *primero el mejor* es la que se utiliza más ampliamente, como principal ventaja destaca que no se evalúan nodos de forma innecesaria. La principal desventaja es la gestión de la lista, ya que el número de nodos puede ser elevado y el impacto en el rendimiento del algoritmo puede ser relevante. Se puede encontrar un estudio más en profundidad sobre la estrategia *primero el mejor* en [13, 14].

Además de estas estrategias, se han estudiado otras en el ámbito de los algoritmos de optimización global intervalar. L.G. Casado propuso en [17] una estrategia de selección basada en un valor heurístico, llamado $p\overline{f^*}(X)$:

$$p\overline{f^*}(X) = \frac{\overline{f^*} - \underline{F}(X)}{\overline{F}(X) - \underline{F}(X)}. \tag{3.1}$$

El valor de $p\overline{f^*}(X)$ relaciona la anchura de la caja con la distancia entre el límite inferior y la cota superior del mínimo, demostrando ser una buena heurística para determinar la proximidad de una caja al mínimo. Esta heurística ha sido ampliamente estudiada en otras investigaciones, véase [20, 21, 23, 30, 31, 68, 76].

3.3.2 Estudios sobre la regla de división

La regla de división consiste en particionar una caja X en un número finito de subcajas X^i que satisfaga la condición $X = \cup X^i$. Uno debe tener en cuenta dos aspectos cuando divide un nodo o caja: la dirección o coordenada de la caja a dividir y el número de subcajas a generar.

Uno podría considerar, además, realizar divisiones de manera no uniforme, esto es, que las cajas generadas tras la división no tuvieran el mismo tamaño y/o forma.

3.3.2.1 Escoger la dirección para la división

En [32] se compara la eficiencia de cinco estrategias para elegir la mejor coordenada por la que dividir una caja. Estas estrategias se deben a investigaciones de Hansen ([53]) y Ratz ([119]). Todas ellas seleccionan la coordenada con una función de mérito:

$$\arg \max_{i=1}^{n} D(i), \tag{3.2}$$

donde $D(i)$ es determinado por cada estrategia. A continuación describimos las funciones de mérito para cada una de las estrategias y el objetivo que se persigue en cada una de ellas.

Estrategia A. Esta estrategia selecciona la coordenada más ancha de la caja.

$$D(i) = w(X_i). \tag{3.3}$$

Es la estrategia más ampliamente usada y la mejor opción cuando no hay disponible información sobre la derivada de la función objetivo [118].

Estragegia B. Hansen [53] describió esta regla. El objetivo es seleccionar la coordenada i donde f tiene su mayor variación, estimado a través de

$$D(i) = w(F_i'(X)) \cdot w(X_i). \tag{3.4}$$

Estrategia C. La idea principal de esta estrategia, propuesta por Ratz [119], es minimizar el exceso de anchura (véase la sección 2.5.1, página 18). Se formula como

$$D(i) = w(F_i'(X)) \cdot (X_i - m(X_i)). \tag{3.5}$$

Estrategia D. Esta estrategia, que no necesita información sobre la derivada (similar a la estrategia A), también trata de minimizar el exceso de anchura. Su formulación es

$$D(i) = \begin{cases} w(X_i) & \text{si } 0 \in X_i, \\ w(X_i)/ < X_i > & \text{en otro caso,} \end{cases} \tag{3.6}$$

donde $< X >$ es la *mignitud* del intervalo X: $< X >= \min |X_i|$.

Estrategia E. Similar a la estrategia C, pretende minimizar la anchura de la inclusión, pero esta vez con información de la segunda derivada (véase [120]).

$$D(i) = w((X_i - m(X_i)) \cdot (F_i'(m(X))) + \frac{1}{2} \sum_{j=1}^{n} (H_{ij}(X) \cdot (X_i - m(X_i))). \tag{3.7}$$

Los resultados experimentales analizados en [32] muestran que las estrategias B, C y E son significativamente mejores que las estrategias A y D, y más significativas cuanto mayor es la dificultad de la función objetivo. La estrategia C es la que se muestra mejor en la mayoría de los casos, seguida por la B y la E, en ese orden.

3.3.2.2 Número de subcajas a generar

La estrategia de división más generalizada es la bisección ($s = 2$), es decir, dividir la caja X en dos subcajas. La elección de la coordenada a dividir puede realizarse según las estrategias descritas en la sección anterior.

Si la estrategia de división genera más de dos cajas, existen dos formas de realizar esta división múltiple: (a) multisección y (b) multicorte. La multisección consiste en aplicar varias veces bisección ($s > 2$) mientras que el multicorte consiste en realizar más de una división en la dimensión seleccionada.

La idea de la multisección surgió en varios trabajos [13, 14, 53, 119]. Un estudio más reciente [29, 88] trata de determinar las implicaciones teóricas y prácticas en cuanto a la mejora de rendimiento al usar multisección y multicorte frente a la estrategia clásica de la bisección. Las conclusiones del estudio muestran que aplicar multicorte empeora el rendimiento del algoritmo. En cuanto a la multisección, utilizarla mejora sustancialmente la eficiencia, en el mejor caso un $20 - 22\,\%$ respecto al algoritmo con bisección, cuando el problema a resolver es complejo. Este estudio también reafirma los resultados presentados en cuanto a la dirección a escoger para la subdivisión, ya que la estrategia C presenta los mejores resultados, para valores de $s = 3$ y $s = 4$.

En [19] se presenta un estudio interesante sobre cómo realizar multisección de forma eficiente. El estudio se basa en aplicar diferentes estrategias de división en función del valor del $p\overline{f^*}(X)$ de la caja en cuestión. Las estrategias utilizadas fueron: (a) bisección, (b) generar 2^n subcajas de tamaño la mitad que su predecesora, (c) generar 3^n y (d) 4^n subcajas, respectivamente. Además, el estudio utilizaba dos parámetros como valores umbrales, P_1 y P_2, de modo que si el valor del $p\overline{f^*}(X) < P_1$ se aplicaría la estrategia (a), si el valor $P_1 \leq p\overline{f^*}(X) < P_2$ se aplicaría la estrategia (b) y si el valor de $p\overline{f^*}(X) > P_2$ se aplicaría la estrategia (d). Los parámetros P_1 y P_2 se optimizaron de forma global para todas las funciones test (a esta estrategia se la llamó PA) e individualmente para cada una de las funciones test (llamada estrategia P). Estas dos nuevas estrategias de multisección obtuvieron los mejores resultados de rendimiento frente a aplicar las estrategias (a), (b), (c) y (d) individualmente.

Por tanto, de los estudios anteriores podemos concluir que la multisección puede ser relevante para mejorar la eficiencia de los algoritmos de optimización global.

3.3.3 Estudios sobre la regla de acotación

Las funciones de inclusión mediante las formas centradas, ya mencionadas en la sección 2.5.1, suelen usarse con más frecuencia en los algoritmos de optimización global intervalar debido a su convergencia cuadrática [69]. En esta sección mostraremos otros estudios en este ámbito.

3.3.3.1 Función de inclusión mediante pendientes

El uso de funciones de inclusión mediante pendientes fue propuesto por Ratz [119] en 1998.

Definición 10 *Una función* $sf : \mathbb{R}^n \to \mathbb{R}^n$ *con respecto a un punto* m *se define como en forma de pendiente (slope form) tal que*

$$f(x) - f(m) = sf(x, m)(x - m)\ \forall x.$$

Si SF es una función de inclusión de sf entonces

$$f(X) \subseteq f(m) + SF(X, m) \cdot (X - m).$$

Se puede encontrar más información sobre funciones de inclusión con forma de pendiente en [105, 123]. Una ventaja de la forma mediante pendientes con respecto a la forma del valor medio y la forma de Taylor es que no requiere que f sea diferenciable. Otros trabajos que siguen las ideas propuestas por Ratz [122] son [10, 24, 90, 134].

3.3.3.2 Forma afín

La aritmética afín[1] fue introducida por Comba, Stolfi y Figueiredo [26, 34, 35, 36] y es un modelo de aritmética verificada que trata de mejorar la aritmética de intervalos. En la forma afín, cada variable x es representada por la fórmula

$$\hat{x} = x_0 + x_1\epsilon_1 + x_2\epsilon_2 + \ldots + x_n\epsilon_n,$$

donde los coeficientes x_i son números en punto flotante conocidos y ϵ_i son variables simbólicas cuyos valores solo se sabe que están en el intervalo $[-1, 1]$.

[1] http://www.ic.unicamp.br/∼stolfi/EXPORT/projects/affine-arith/

Ejemplo 6 *Imaginemos una variable x que debe estar en el intervalo $[3, 7]$, puede ser representada por su forma afín $\hat{x} = 5 + 2\epsilon_k$, para algún k. A la inversa, la forma $\hat{x} = 10 + 2\epsilon_3 - 5\epsilon_8$ implica que el valor de la variable x está incluido en el intervalo $[3, 17]$.*

Si dos formas afines $\hat{x}$, $\hat{y}$ comparten una variable ϵ_k, significa que las correspondientes variables x e y son parcialmente dependientes, en el sentido de que su rango conjunto es menor que el producto cartesiano de sus intervalos por separado.

Ejemplo 7 *Sea $\hat{x} = 10 + 2\epsilon_3 - 6\epsilon_8$ e $\hat{y} = 20 + 3\epsilon_4 + 4\epsilon_8$, entonces sus rangos individuales son $x \in [2, 18]$ e $y \in [13, 27]$, respectivamente. Sin embargo, el rango del par $(\hat{x}, \hat{y})$ es un hexágono con los vértices $(2, 27)$, $(6, 27)$, $(18, 19)$, $(18, 13)$, $(14, 13)$ y $(2, 21)$; que es un subconjunto del rectángulo $[2, 18] \times [13, 27]$.*

Las operaciones de la aritmética afín entre dos formas afines proveen un resultado afín. Para operaciones no afines se debe utilizar una aproximación afín y el posible error de la aproximación es representado en un nuevo término extra ϵ_k que no ocurre en ninguna forma afín previa.

Frédéric Messine extendió la aritmética afín mediante la incorporación de intervalos para su aplicación en la optimización global, usándola como función de inclusión [91, 94]. Los resultados obtenidos muestran que la aritmética afín reduce el problema de la *dependencia de intervalos* de la aritmética de intervalos, mejorando los límites inferiores y superiores obtenidos. Los resultados muestran que los nuevos métodos obtienen un orden de convergencia cuadrático sin necesidad de calcular una inclusión del grandiente.

3.3.4 Estudios sobre la regla de eliminación

La regla de eliminación es una pieza clave para el rendimiento del algoritmo de optimización. Uno puede caer en la tentación (erróneamente) de intentar aplicar el mayor número de mecanismos de rechazo, pero esta decisión puede repercutir negativamente en la eficiencia debido al coste que requiere la evaluación de cada test de rechazo.

En la literatura existen algunas técnicas heurísticas de rechazo de cajas que no garantizan la rigurosidad de la solución [20, 21, 23]. Además, podemos distinguir

dos tipos de reglas de eliminación: aquellas que eliminan una caja completa puesto que se ha probado que en su interior no está la solución, y aquellas que permiten reducir el tamaño de la caja [24, 54, 79, 89, 90, 131]. A continuación, describiremos las reglas de eliminación más relevantes existentes en la bibliografía.

3.3.4.1 Test de rango y test de corte

El test de rango es la principal y más sencilla herramienta de rechazo de cajas. Dada una cota superior del mínimo denotada por $\overline{f^*}$, aquellas cajas cuyo límite inferior del rango real esté por encima de este valor, $\underline{F}(X) > \overline{f^*}$, pueden ser eliminadas con la certeza de que en su interior no se encuentra el óptimo global. Si durante la ejecución del algoritmo el valor de $\overline{f^*}$ es mejorado, se puede comprobar si las cajas guardadas en la lista de trabajo o en la lista de nodos solución pueden ser rechazadas debido a esta nueva cota superior del mínimo. A este procedimiento se le conoce como test de corte.

Por otro lado, si una caja no puede ser rechazada mediante el test de rango, es posible reducir el tamaño de la misma utilizando el valor de $\overline{f^*}$, eliminando las regiones en las que $F(X) > \overline{f^*}$. En [54] se describen cuatro métodos para lograr esto.

3.3.4.2 Test de monotonía y *hull consistency*

Asumiremos que la función f es una función diferenciable y que F' es la función de inclusión del gradiente de f. El objetivo es encontrar los valores que hacen cero el gradiente ya que son mínimos, máximos o puntos de inflexión.

El test de monotonía fue introducido por Hansen [52] para probar que no existen puntos estacionarios en el interior de un intervalo, en tal caso, el intervalo puede ser rechazado. La propiedad 4 definida anteriormente (página 20) describe el test de monotonía para intervalos.

Otro procedimiento que podemos utilizar para descartar cajas que no contienen puntos estaciones es el *hull consistency* [54]. Este procedimiento tiene similar coste computacional y además permite reducir el tamaño de la caja.

3.3.4.3 Test de concavidad o convexidad

Asumiremos que la función f es dos veces diferenciable. La región próxima al mínimo global x^* debe ser convexa. Por tanto, la matriz Hessiana H de f debe ser semidefinida positiva [5]. Una condición necesaria para esto es que los elementos de la diagonal principal de H sean no negativos, esto es, si $\overline{H}_{ii}(X) < 0$ para algún $i = 1, \ldots, n$, entonces X no puede contener un mínimo y puede ser eliminada. Hay otras condiciones necesarias para que H sea semidefinida positiva [5], pero evaluarlas todas puede influir negativamente en el rendimiento del algoritmo.

3.3.4.4 Test de Newton

Isaac Newton creó un método para resolver una ecuación de la forma $f(x) = 0$ con $f : \mathbb{R} \to \mathbb{R}$. El principal problema es que dependiendo del punto inicial escogido, el método puede no converger a una solución. La extensión a intervalos del método de Newton fue introducida por Moore [97] y carece de estas deficiencias. La extensión a intervalos del método de Newton unidimensional, $N(X)$ viene definido por la siguiente ecuación:

$$X_{n+1} = N(X) \cap X_n, \text{ hasta que } X_{n+1} = X_n \text{ o } X_{n+1} = \emptyset, \tag{3.8}$$

donde $N(X) = a - \frac{f(a)}{F'(X)},\ a \in X$. Normalmente $a = m(X)$. Si $0 \in F'(X)$, entonces $N(X) \cap X_n$ dará como resultado dos intervalos semi-infinitos. Este método tiene las siguientes propiedades [53, 97, 118]:

- Cualquier raíz $b \in X_n$ de $f(x)$ cumple que $b \in N(X_n)$.
- Si $N(X_n) \cap X_n = \emptyset$, entonces no existe una raíz de $k(x)$ en X.
- Si $N(X_n) \subset X_n$, entonces existe una sola raíz de $f(x)$ en $N(X_n)$.
- Si existe una raíz de $f(x)$ en X, entonces la convergencia, en términos de la anchura del intervalo, es cuadrática, siempre que los errores de redondeo computacionales sean menores que la precisión final requerida.

La ecuación (3.8) puede generalizarse al caso multidimensional, estableciendo el vector $x_n = m(X)$ así:

$$N(X_{n+1}) = x_n - [F'(X_n)]^{-1} f(x_n),\ x_n \in X_n. \tag{3.9}$$

Para obtener $N(X_{n+1})$ hay que resolver el siguiente sistema de ecuaciones:

$$F'(X)(N(X_{n+1}) - x_n) = -f(x_n). \tag{3.10}$$

La resolución de la ecuación (3.10) puede ser compleja. Los principales métodos que existen son el método de Krawczyk [99, 100] y el método de Gauss-Seidel extendido a intervalos [70].

El test de Newton se utiliza en los problemas de optimización para comprobar si una caja contiene un máximo, un mínimo o un punto de inflexión, resolviendo la ecuación $f'(x) = 0$. Para ello la función $f(x)$ debe ser dos veces diferenciable.

3.3.5 Estudios sobre la regla de finalización

Se suelen utilizar dos criterios para considerar que un nodo pertenece a la lista de nodos solución. El primero de ellos se basa en que la anchura de la caja alcance una precisión prefijada, $w(X) \leq \epsilon_X$. El segundo criterio se basa en que la anchura del intervalo de inclusión alcance una precisión prefijada, $w(F(X)) \leq \epsilon_f$. Es posible combinar ambos criterios ([54]) de forma tal que una caja deba cumplir uno de ellos o los dos, para incluirla en la lista de soluciones. Si no estamos interesados en obtener los puntos x^* donde se obtiene el mínimo global, es decir, solo nos interesa el valor del mínimo f^*, podemos modificar el criterio de finalización y rechazar aquellas cajas donde $\underline{F}(X) > \overline{f^*} - \epsilon_f$. En tal caso, el mínimo estará en el intervalo $[\overline{f^*} - \epsilon_f, \overline{f^*}]$.

Estudia el pasado si quieres pronosticar el futuro.

Confucio

CAPÍTULO

4

Predicción de la carga computacional

4.1 Introducción

El tiempo de cómputo necesario para resolver cualquier tipo de problema puede reducirse mediante computación paralela. Sin embargo, para conseguir un buen rendimiento, el algoritmo paralelo debe administrar adecuadamente la carga de trabajo entre los procesadores. Para lograr esto, es necesario estimar la carga de trabajo real y futura de cada problema para poder determinar los recursos computacionales que son necesarios en cada momento de la ejecución.

En esta introducción revisaremos los métodos para estimar la carga de trabajo en los algoritmos de ramificación y acotación. Para una amplia revisión de algoritmos paralelos de ramificación y acotación véase [46, 133]. En algoritmos con múltiples listas de trabajo, la mayoría de los autores discute la prevención de anomalías en la búsqueda paralela. La estrategia utilizada para lograrlo es realizar búsquedas de modo similar a los algoritmos secuenciales, moviendo los nodos más prometedores (aquellos con mejores límites inferiores en problemas de minimización, véase la sección 3.3.1, página 30) entre procesadores. Algunos ejemplos pueden encontrarse en [43, 47, 81, 116].

Los algoritmos paralelos de ramificación y acotación se han usado para resolver problemas de optimización global, véase por ejemplo [42, 110, 114]. Ejemplos de algoritmos paralelos de optimización global intervalar pueden encontrarse en [15, 18, 41, 58, 59, 64, 68, 101, 135]. Estas referencias muestran la importancia de conocer la carga de trabajo del algoritmo de ramificación y acotación, pero ninguna de ellas intenta estimar el trabajo pendiente.

Knuth [74] fue el primero en proponer un método de predicción para árboles de *backtracking*. Argumentó que su método *off-line* basado en pruebas aleatorias no podía ser usado en métodos como el de ramificación y acotación. Cornuéjols et al. [28] da algunas razones intuitivas sobre por qué el estimador de Knuth podría fallar, y presenta un método para estimar el tamaño de un árbol de ramificación y acotación en problemas MIP (*Mixed Integer Programming*) usando tres parámetros: la profundidad máxima del árbol, el nivel más ancho y el primer nivel en el que el árbol no es completo. Kilby et al. [73] ofrece una revisión extensa sobre el tema y presenta dos métodos *on-line* para estimar el tamaño de un árbol de *backtracking*. El primer método se basa en una muestra con pesos de las ramas visitadas de forma cronológica. El segundo método usa una búsqueda en profundidad de izquierda a derecha. Asume que el subárbol derecho no explorado será similar al subárbol izquierdo explorado. Özaltın et al. [109] estudian un método basado en monitorizar el *MIP gap* para predecir cuando este *gap* se convierte en cero y la búsqueda finaliza. Los experimentos mostraron que aplicar la suma de los *gaps* de los subárboles ofrece mejores predicciones que usar el *gap* global. En [9] se estudian tres métodos de estimación de la carga computacional en términos de trabajo pendiente, para árboles de ramificación y acotación. Ninguno de esos métodos asume que el árbol deba tener una forma o regularidad determinada. A continuación se describen esos métodos.

4.2 El concepto de *left-over*

En esta sección definiremos los conceptos necesarios que ayuden a entender el problema de predecir la carga computacional pendiente, en términos de nodos que quedan por evaluar en el árbol de ramificación y acotación. A efectos de cálculo, podemos considerar cada uno de estos nodos como el nodo raíz de su subárbol.

Denominaremos como descendientes (*offspring*) de un nodo al número total de nodos del subárbol del que este nodo es raíz y lo definimos de la siguiente manera:

Definición 11 *$Offspr(X)$ es el número de nodos generados en el subárbol del nodo X por un algoritmo de ramificación y acotación.*

Para poder realizar la predicción del trabajo total pendiente en el algoritmo de ramificación y acotación (véase los algoritmos 1 y 2), es necesario conocer el número total de nodos descendientes de los nodos almacenados en el árbol Λ. Este valor lo denominaremos *left-over* y lo definimos tal que así:

Definición 12 *$LeftOver(\Lambda)$ es el número total de descendientes de los nodos almacenados en Λ en un algoritmo de ramificación y acotación:*

$$LeftOver(\Lambda) = \sum_{X \in \Lambda} Offspr(X). \tag{4.1}$$

El número exacto de descendientes de un nodo, $Offspr(X)$, no es conocido hasta que la ejecución del algoritmo finaliza, de ahí la dificultad de conocer el valor de (4.1) durante la ejecución del algoritmo de ramificación y acotación. Sí es posible establecer una cota superior de este valor, lo denominaremos $CTree(X)$:

Definición 13 *$CTree(X)$ es el número de nodos del árbol completo que generaría el nodo X en un algoritmo de ramificación y acotación.*

El valor de $CTree(X)$ representa el número máximo de nodos que tendría que evaluar el algoritmo de ramificación y acotación si no se eliminara ningún nodo, es decir, si se tuviera que evaluar el árbol completo. Este valor depende de la anchura de X ($w(X)$), de la dimensión del problema (n), de la precisión deseada (ϵ) y de la regla de división. En adelante asumiremos que el algoritmo empleado es el algoritmo 2, el cual fue presentado en la sección 3.2 y que usa como regla de división la bisección.

Denotaremos por L a la profundidad máxima del árbol de búsqueda a la que hay que llegar para alcanzar los nodos solución a partir del nodo raíz S. Cada nodo requiere ser dividido n veces para que su anchura sea la mitad. Se puede determinar el número de veces que se ha de reducir a la mitad la anchura de un nodo, comenzando por el nodo raíz S, para obtener nodos solución ($w(X) \leq \epsilon$),

este valor lo denotaremos por m. Asumiremos que la región de búsqueda S tiene un tamaño igual en todas sus dimensiones. El valor de m se determina con

$$\frac{w(S)}{2^m} \leq \epsilon.$$

Por tanto,

$$m = \left\lceil \log_2(\frac{w(S)}{\epsilon}) \right\rceil. \tag{4.2}$$

Entonces, la profundidad máxima del árbol de ramificación y acotación es

$$L = m \cdot n. \tag{4.3}$$

Si la anchura de todas las dimensiones del nodo S ($w_i(S)$) no son iguales,

$$L = \lceil \sum_{i=1}^{n} \log_2 w_i(S) - \log_2 \epsilon \rceil. \tag{4.4}$$

Sea $l(X)$ el nivel del nodo X en el árbol de ramificación y acotación, donde $l(S) = 0$, $CTree(X)$ es el número de nodos del árbol binario completo de profundidad $L - l(X)$:

$$CTree(X) = 2^1 + 2^2 + 2^3 + \cdots + 2^{L-l(X)}. \tag{4.5}$$

También puede escribirse en su forma abreviada como:

$$CTree(X) = 2^{(L-l(X)+1)} - 2 = 2(2^{(L-l(X))} - 1). \tag{4.6}$$

Dada una lista de nodos Λ en un algoritmo de ramificación y acotación, podemos establecer un valor máximo del número de nodos a evaluar por el algoritmo, es decir, una cota superior de $LeftOver(\Lambda)$, que denotaremos por $LeftOver^U(\Lambda)$ y que calcularemos como:

$$LeftOver^U(\Lambda) = \sum_{X \in \Lambda} CTree(X).$$

Ejemplo 8 *Imaginemos que queremos resolver el problema de optimización global conocido como Goldstein-Price [124] para una precisión* $\epsilon = 10^{-3}$. *Este problema tiene dos dimensiones,* $n = 2$, *y el nodo de búsqueda inicial es* $S = [-2, 2]^2$. *Empleando las ecuaciones (4.2) y (4.3) obtenemos que* $L = 2 \cdot \left\lceil \log_2(\frac{4}{10^{-3}}) \right\rceil = 24$.

Dado que el nodo inicial S tiene una profundidad $l(X) = 0$, sustituyendo en (4.6) obtenemos que el número máximo de nodos a evaluar para resolver este problema mediante el algoritmo 2 es $CTree(S) = 33\,554\,430$, más de treinta y tres millones de nodos.

El ejemplo 8 ilustra cómo el número máximo de nodos a evaluar para resolver un problema de optimización global puede ser realmente elevado, aún cuando el problema no es excesivamente complejo (solo tiene dos dimensiones y la precisión de la solución no es muy exigente). Además, el cálculo del número máximo de nodos a evaluar es independiente de la complejidad de la función, puesto que depende de la dimensión del problema, del tamaño del espacio de búsqueda, de la regla de división y de la precisión de la solución. El número de nodos que se evaluarán realmente es difícil de determinar a priori y depende en gran medida de los test de rechazo (véase la sección 3.3.4) y del resto de reglas del algoritmo de ramificación y acotación. Por ejemplo, para el problema mostrado en el ejemplo 8, el algoritmo 2 evaluaría en realidad $101\,668$ nodos.

La figura 4.1 muestra un árbol binario completo de cuatro niveles y el valor de $LeftOver^U(\Lambda)$ con $\Lambda = \{S\}$.

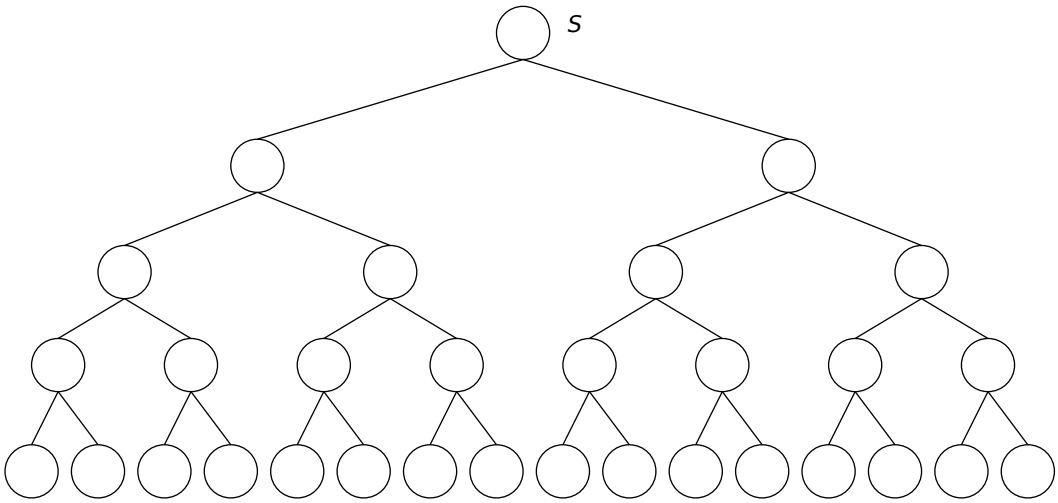

Figura 4.1: Árbol binario completo - Profundidad del árbol $L = 4$, $LeftOver^U(\Lambda) = CTree(S) = 2^1 + 2^2 + 2^3 + 2^4 = 30$.

El valor de $LeftOver^U(\Lambda)$ es un valor muy elevado y se aleja demasiado del valor real del número de nodos, $LeftOver(\Lambda)$. Este valor depende de la efectividad

de las reglas de rechazo, por lo que es necesario estimar el comportamiento de estas reglas para obtener una buena estimación del $LeftOver(\Lambda)$. Las figuras 4.2 y 4.3 simulan el comportamiento de un árbol de ramificación y acotación para ejemplificar la dificultad de estimar estos valores. Los nodos por encima de la línea punteada representan lo que ha sucedido hasta ese momento de la ejecución, mientras que los nodos por debajo de esta línea representan lo que pasará en el futuro y, por tanto, es desconocido en ese punto de la ejecución. Los nodos de color azul representan los nodos almacenados en Λ (en el momento actual de la ejecución), los nodos de color blanco son aquellos que han sido o serán seleccionados y divididos, los nodos de color gris representan aquellos nodos que han sido o serán rechazados y por último, los nodos de color rojo representan los nodos solución.

La figura 4.2 representa el estado del árbol en un momento determinado de la ejecución. En este momento, la diferencia entre el valor de $LeftOver(\Lambda)$ y de $LeftOver^U(\Lambda)$ es de 12.

A medida que la ejecución del algoritmo avanza, la cota superior del *left-over* se acerca a su valor real. La figura 4.3 muestra cómo se ha reducido esta diferencia a 6 desde el punto de la ejecución mostrado en la figura 4.2, debido al rechazo de nodos con un valor sobrestimado de sus descendientes.

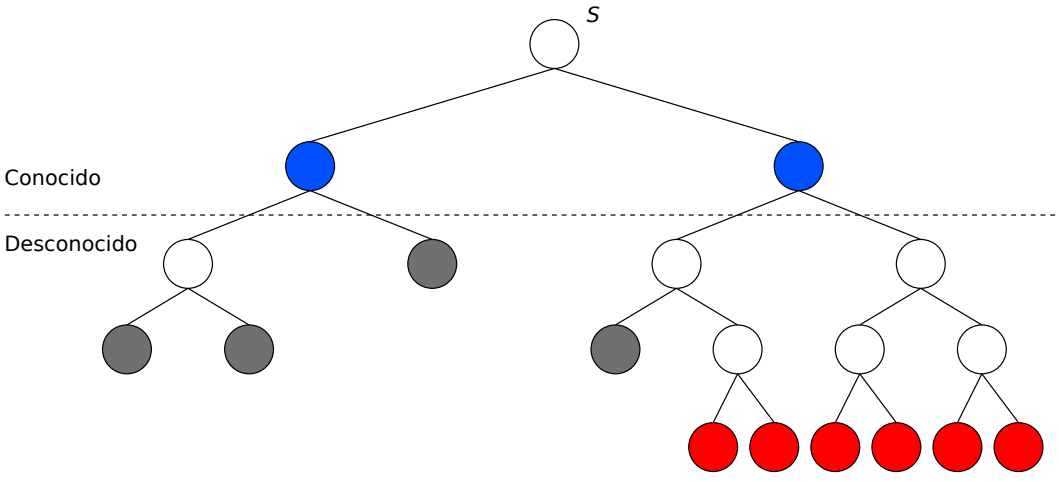

Figura 4.2: Evolución de un árbol de ramificación y acotación. Primera instantánea - Profundidad $L = 4$, $LeftOver(\Lambda) = 16$ y $LeftOver^U(\Lambda) = 28$.

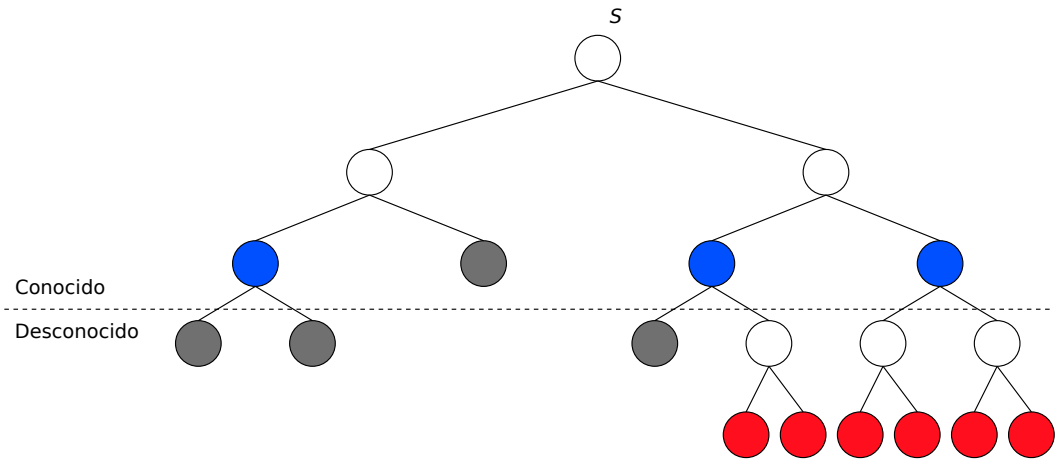

Figura 4.3: Evolución de un árbol de ramificación y acotación. Segunda instantánea - Profundidad $L = 4$, $LeftOver(\Lambda) = 12$ y $LeftOver^U(\Lambda) = 18$.

Las figuras 4.2 y 4.3 ilustran la necesidad de estimar el número de descendientes de un nodo X, a este valor lo denotaremos por $Offspr^E(X)$. Para estimar el valor de *left-over*, nuestra aproximación pasa por estimar el valor de los descendientes de cada uno de los nodos en Λ.

Definición 14 *El valor estimado de left-over, denotado por $LeftOver^E(\Lambda)$, en un algoritmo de ramificación y acotación, es la suma de los valores estimados de los descendientes de todos los nodos en Λ:*

$$LeftOver^E(\Lambda) = \sum_{X \in \Lambda} Offspr^E(X) \tag{4.7}$$

El objetivo es diseñar métodos que obtengan un valor de $Offspr^E(X)$ tan cercano al valor real, $Offspr(X)$, como sea posible. Estos métodos deben ser sencillos de calcular para no influir negativamente en el rendimiento del algoritmo. Una posible aproximación es observar la cantidad de nodos rechazados por el algoritmo en cada iteración. En la siguiente sección estudiamos este caso.

4.3 Cálculo del factor de rechazo

El factor de rechazo del algoritmo de ramificación y acotación se utiliza para obtener una buena estimación del número de descendientes de un nodo, lo que hemos

denotado por $Offspr^E(X)$. Hay diferentes maneras de medir este factor de rechazo, nosotros consideraremos las siguientes:

1. Midiendo el número de nodos eliminados en cada nivel del árbol.

2. Midiendo el número de nodos eliminados en las iteraciones previas del algoritmo.

Consideraremos que las predicciones se realizan cada k iteraciones del algoritmo y usaremos j como el índice para cada predicción, de tal modo que cada $j \times k$ iteraciones se realiza una predicción. A continuación detallamos cada uno de estos métodos.

4.3.1 Factor de rechazo basado en niveles

El factor de rechazo en un nivel del árbol es la razón entre el número de nodos rechazados y el número de nodos evaluados en ese nivel. Usaremos el subíndice i para denotar el nivel del árbol.

Definición 15 *El factor de rechazo γ_i por nivel $i = 1, \ldots, L$; es el número de nodos rechazados en el nivel i dividido por el número de nodos evaluados en ese mismo nivel. La serie de números es llamada secuencia-γ.*

La secuencia-γ se puede emplear para calcular el número total de nodos del árbol de ramificación y acotación desde el nodo raíz. La ecuación (4.8) muestra cómo se realiza este cálculo.

$$Offspr(S) = 2 \cdot \sum_{i=2}^{L} (2^i \cdot \prod_{p=1}^{i-1} (1 - \gamma_p)). \tag{4.8}$$

con $\gamma_0 = 0$.

La figura 4.4 ilustra el concepto de la secuencia-γ en un árbol de cuatro niveles. Conocidos los valores de γ_i, podemos deducir el número total de nodos del árbol aplicando la ecuación (4.8):

$$Offspr(S) = 2^1 + 2^2(1 - \gamma_1) + \cdots + 2^4(1 - \gamma_1)(1 - \gamma_2)(1 - \gamma_3) = 18. \tag{4.9}$$

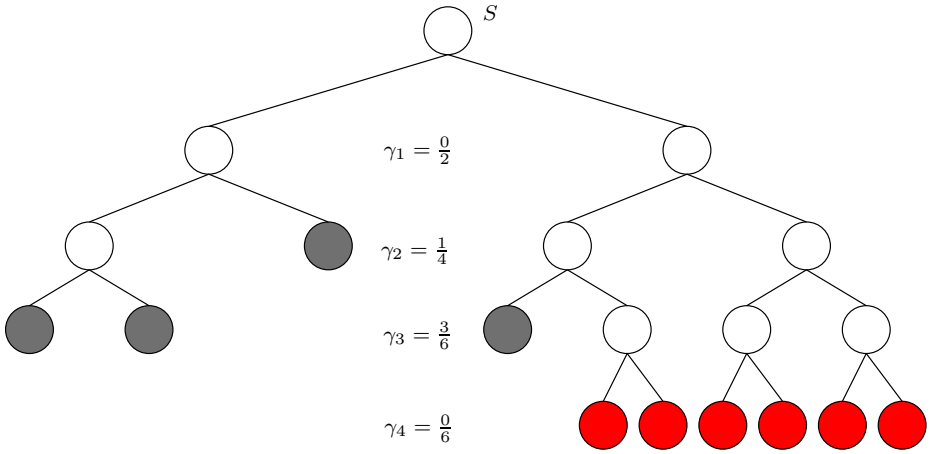

Figura 4.4: Factores de rechazo por niveles y secuencia-γ - Obtención de los factores de rechazo por niveles de un árbol de ramificación y acotación para el cálculo del número total de nodos del árbol, $Offspr(S) = 18$.

Usaremos la idea mostrada en la ecuación (4.8) para estimar el número de nodos del árbol. Para ello, es necesario tener una buena estimación de la secuencia-γ.

Conociendo los valores exactos de la secuencia-γ, conoceremos el número total de nodos que se evaluarán en el árbol de ramificación y acotación. La dificultad reside en que no se conocen los valores exactos de la secuencia-γ hasta que el algoritmo finaliza, y una pequeña variación en el valor de los mismos puede hacer que la estimación sea bastante errónea. Introduciremos varios conceptos para estimar el factor de rechazo por nivel:

$E_{i,j}$: número de nodos evaluados en el nivel i tras $j \times k$ iteraciones.
$R_{i,j}$: número de nodos rechazados en el nivel i tras $j \times k$ iteraciones.
q_j : nivel más bajo para el que $E_{i,j} = 0,\ \forall i \geq q_j$. Cuando el algoritmo finaliza $q_j = L + 1$.

El valor de q_j representa la profundidad actual del árbol, ya que es el nivel para el que aún no se ha evaluado ninguna caja. Estos conceptos nos permiten definir el factor de rechazo observado.

Definición 16 *El factor de rechazo observado por nivel i durante la ejecución del algoritmo tras $j \times k$ iteraciones es:*

$$\hat{\gamma}_{i,j} = \frac{R_{i,j}}{E_{i,j}}, i = 1,\ \ldots,\ q_j - 1. \tag{4.10}$$

La serie de números la denotaremos por secuencia-$\hat{\gamma}_j$.

Nótese que para aquellos niveles en los que todos los nodos han sido evaluados, $\gamma_i = \hat{\gamma}_{i,j}$. La figura 4.5 muestra un ejemplo donde $q_1 = 3$, $E_{2,1} = 2$ y $R_{2,1} = 1$. Por tanto, el reto al que nos enfrentamos es obtener buenas estimaciones de los valores desconocidos (γ_3 y γ_4 en la figura 4.5) y parcialmente conocidos (γ_2 en la figura 4.5) de γ_i.

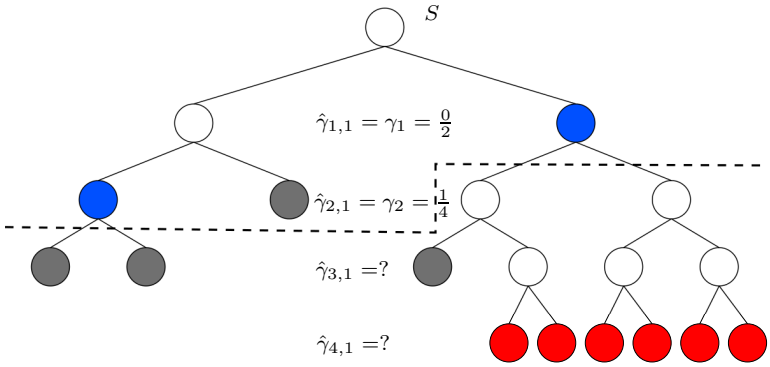

Figura 4.5: Factores de rechazo no conocidos por niveles - Los factores de rechazo por niveles son parcialmente conocidos o desconocidos durante la ejecución del algoritmo de ramificación y acotación. El valor $q_1 = 3$ representa el nivel más profundo del árbol para el que aún no se ha evaluado ningún nodo.

4.3.2 Factor de rechazo basado en iteraciones

Otro método para estimar cómo se rechazan los nodos en el árbol es medir la cantidad de nodos rechazados en un número determinado de iteraciones. Nos centraremos en medir el comportamiento de las últimas k iteraciones. A continuación, definimos varios conceptos siguiendo una notación similar a la realizada en la sección 4.3.1:

EI_j : número de nodos evaluados en las últimas k iteraciones. Es un valor constante que depende de la regla de división, por ejemplo, si usamos bisección $EI_j = 2k$.

RI_j : número de nodos rechazados en las últimas k iteraciones.

FI_j : numero de nodos finales almacenados en Ω (véase el algoritmo 2) en las últimas k iteraciones.

Definición 17 *El factor de rechazo por iteraciones, denotado por φ_j, es el cociente entre el número de nodos rechazados más el número de nodos finales, y el número de nodos evaluados en las últimas k iteraciones de un total de $j \times k$ iteraciones:*

$$\varphi_j = \frac{RI_j + FI_j}{EI_j} \in [0, 1]. \tag{4.11}$$

Un valor de $\varphi_j > 0.5$ significa que en su mayoría los nuevos nodos son rechazados, por lo que el tamaño de Λ se reduce. Un valor de $\varphi_j < 0.5$ significa que en su mayoría los nuevos nodos son almacenados, por lo que el tamaño de Λ aumenta. Un valor de $\varphi_j = 0.5$ significa que se rechaza y se almacena el mismo número de nodos, por lo que el tamaño de Λ permanece igual. Se puede aplicar un razonamiento similar para $\hat{\gamma}_{i,j}$ en la sección 4.3.1.

4.4 Estimación de la profundidad de un subárbol

En general, no todos los nodos alcanzan el nivel final en el árbol de ramificación y acotación (véase la figura 4.2) ya que pueden ser rechazados mucho antes. Por tanto, una estimación del nivel máximo del árbol que alcanzará un nodo puede ayudar a mejorar la estimación de los descendientes de ese nodo (véase la definición 14).

Sea X^P el nodo padre de X y X^A el nodo abuelo de X (el nodo padre de X^P) entonces, por la propiedad de isotonía de las funciones de inclusión de intervalos (véase la sección 2.5.1) tenemos que

$$\underline{F}(X^A) \leq \underline{F}(X^P) \leq \underline{F}(X).$$

Esto se ilustra en la figura 4.6. Se pueden usar los incrementos de los valores del límite inferior entre los nodos X^A, X^P y X para estimar la profundidad del subárbol de X. Suponiendo que el factor de reducción del límite inferior se mantiene constante para los nodos descendientes de X, podemos trazar dos pendientes, la primera con los valores de $\underline{F}(X^A)$ y $\underline{F}(X^P)$; y la segunda con los valores de $\underline{F}(X^P)$ y $\underline{F}(X)$. El nivel máximo en el que las pendientes cortan con el valor de $\overline{f^*}$ será la profundidad estimada en la que desaparezcan los descendientes de X. La

idea es que en ese momento, el valor del límite inferior de los descendientes de X será mayor que la cota superior del mínimo $\overline{f^*}$ y por tanto, no habría más nodos en el subárbol.

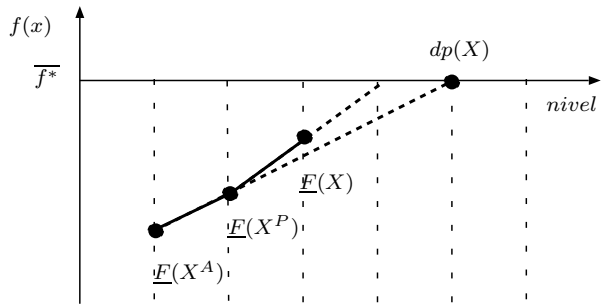

Figura 4.6: Predicción de la profundidad del subárbol de un nodo - La profundidad del subárbol de X viene determinada por $dp(X)$ que estima cuándo los límites inferiores de los descendientes de X serán mayores que $\overline{f^*}$.

El nivel del árbol en el que las pendientes alcanzarán el valor de $\overline{f^*}$ se puede obtener con las ecuaciones (4.12) y (4.13):

$$CutLevel_1(X) = \left\lceil \frac{\overline{f^*} - \underline{F}(X)}{\underline{F}(X) - \underline{F}(X^P)} \right\rceil + l(X), \tag{4.12}$$

$$CutLevel_2(X) = \left\lceil \frac{\overline{f^*} - \underline{F}(X^P)}{\underline{F}(X^P) - \underline{F}(X^A)} \right\rceil + l(X) - 1. \tag{4.13}$$

El nivel máximo estimado para el subárbol de X será el máximo de los valores anteriores, siempre y cuando no se supere la profundidad máxima del árbol de ramificación y acotación, L:

$$CutLevel(X) = \text{mín}\{\text{máx}(CutLevel_1(X), CutLevel_2(X)), L\}.$$

Es decir, si alguna de las pendientes es 0 o si los valores de las ecuaciones (4.12) y (4.13) son mayores que L, entonces $CutLevel(X) = L$.

El número de niveles estimado del subárbol generado por X es:

$$dp(X) = CutLevel(X) - l(X). \tag{4.14}$$

Nótese que $dp(X)$ predice la profundidad del subárbol generado por el nodo X, pero no ofrece información sobre el número de descendientes de X o de cómo estos

descendientes se distribuyen en el árbol. Por tanto, es necesario combinar el valor de $dp(X)$ con los factores de rechazo estimados para obtener una predicción del número de descendientes de X. El valor de $dp(X)$, una vez combinado con el factor de rechazo, sí que ofrece información sobre la carga de trabajo pendiente en cada nodo individualmente, por lo que se podría utilizar para decidir qué nodos mover a otros procesadores en las versiones paralelas de los algoritmos de ramificación y acotación.

4.5 Métodos de estimación del *left-over*

Los factores de rechazo basados en niveles y en iteraciones, $\hat{\gamma}_{i,j}$ y φ_j, y la predicción de la profundidad del subárbol de un nodo, $dp(X)$, pueden usarse para diseñar métodos que estimen el valor de los descendientes de un nodo y del valor del *left-over*. En esta tesis estudiaremos tres métodos diferentes, uno basado en el factor de rechazo por niveles y dos basados en el factor de rechazo por iteraciones. A continuación se detallan cada uno de estos métodos.

4.5.1 Método de estimación del *left-over* basado en niveles (PLLEM)

El número de nodos que quedan por evaluar en el algoritmo de ramificación y acotación es la diferencia entre el número de descendientes del nodo inicial, $Offspr(S)$, y el número de nodos que ya se han evaluado, $E_{i,j}$. Sea Λ_j el estado de la lista de trabajo en la predicción j (tras $j \times k$ iteraciones):

$$LeftOver(\Lambda_j) = Offspr(S) - \sum_{i=1}^{L} E_{i,j}. \tag{4.15}$$

Nótese que usando bisección, $\sum_{i=1}^{L} E_{i,j} = 2 \times k \times j$. El valor de $Offspr(S)$ es desconocido durante la ejecución del algoritmo, pero podemos usar la ecuación (4.8) con los valores observados de secuencia-$\hat{\gamma}_j$ para estimar su valor:

$$Offspr_j^E(S) = 2^1 + 2^2(1 - \hat{\gamma}_{1,j}) + \ldots + 2^L \prod_{i=1}^{L-1}(1 - \hat{\gamma}_{i,j}).$$

Nótese además que el valor de $\hat{\gamma}_{i,j}$ solo ha sido medido en los niveles donde se han evaluado nodos, esto es, $i = 1, \ldots, q_j - 1$. Para el resto de niveles usamos un valor por defecto, $\hat{\gamma}_{i,j} = 0.5$ para $i = q_j, \ldots, L$.

El valor estimado del *left-over* basado en niveles, denotado por PL-LEM, es:

$$LeftOver^E(\Lambda_j) = Offspr_j^E(S) - \sum_{i=1}^{L} E_{i,j}. \tag{4.16}$$

4.5.2 Método de estimación global del *left-over* basado en iteraciones (IG-LEM)

Este método utiliza el factor de rechazo por iteraciones φ_j descrito en la sección 4.3.2. Para el cálculo de φ_j solo se tenían en cuenta las últimas k iteraciones. Usaremos una media ponderada de los valores previos de φ_j para calcular el factor de rechazo de la ejecución, denotado por θ_j:

$$\theta_j = \alpha\theta_{j-1} + (1-\alpha)\varphi_j \in [0,1]. \tag{4.17}$$

Inicialmente $\theta_0 = \varphi_0$. El valor de α debe estar entre $[0,1]$, usaremos $\alpha = 0.4$ porque muestra un buen compromiso entre la historia pasada y los cambios recientes. El último valor obtenido para θ_j se usa para estimar el valor de $Offspr_j^E(X)$. Para ello, incorporamos θ_j al cálculo de $CTree(X)$ en la ecuación (4.5):

$$Offspr_j^E(X) = 2^1 + 2^2(1-\theta_j)^1 + 2^3(1-\theta_j)^2 + \cdots + 2^{L-l(X)}(1-\theta_j)^{L-l(X)-1}$$

que podemos simplificar como:

$$Offspr_j^E(X) = 2\frac{2^{L-l(X)}(1-\theta_j)^{L-l(X)} - 1}{2(1-\theta_j) - 1}. \tag{4.18}$$

Combinado con la ecuación (4.7) obtenemos el estimador del *left-over* IG-LEM:

$$LeftOver^E(\Lambda_j) = \sum_{X\in\Lambda_j} Offspr_j^E(X) = 2\sum_{X\in\Lambda_j} \frac{2^{L-l(X)}(1-\theta_j)^{L-l(X)} - 1}{2(1-\theta_j) - 1}. \tag{4.19}$$

Nótese que la ecuación (4.18) considera que un nodo X tendrá descendientes en todos los niveles del árbol, cuya profundidad máxima es L.

4.5.3 Método de estimación local del *left-over* basado en iteraciones (IL-LEM)

El método IL-LEM es similar a IG-LEM, pero ahora la estimación de la profundidad del subárbol de cada uno de los nodos (véase la ecuación (4.14)) es tenida en cuenta para la estimación de los descendientes de los mismos. Modificando la ecuación (4.18) en consecuencia, obtenemos:

$$Offspr_j^E(X) = 2\frac{2^{dp(X)}(1-\theta_j)^{dp(X)}-1}{2(1-\theta_j)-1}. \tag{4.20}$$

Combinado otra vez con la ecuación (4.7) y con la estimación del número de descendientes teniendo en cuenta la profundidad del subárbol de cada uno de ellos (ecuación (4.20)), obtenemos el estimador del *left-over* IL-LEM:

$$LeftOver^E(\Lambda_j) = \sum_{X\in\Lambda_j} Offspr_j^E(X) = 2\sum_{X\in\Lambda_j} \frac{2^{dp(X)}(1-\theta_j)^{dp(X)}-1}{2(1-\theta_j)-1}. \tag{4.21}$$

Este método necesita más cálculos que IG-LEM para determinar $dp(X)$ para cada nodo. Además, es necesario almacenar los valores de los límites inferiores del nodo padre y del nodo abuelo en cada nodo. Nótese que para diferentes nodos con valores similares de $dp()$ se obtendrán valores iguales de $Offspr^E()$, independientemente del nivel del árbol en el que se encuentren.

4.6 Evaluación experimental de los métodos de estimación

En primer lugar discutiremos las características de los experimentos diseñados y después mostraremos y discutiremos los resultados numéricos obtenidos.

4.6.1 Diseño de los experimentos

Se utilizará el algoritmo 2 para evaluar los métodos PL-LEM (basado en (4.16)), IG-LEM (basado en (4.19)) e IL-LEM (basado en (4.21)). Para medir la calidad de los métodos se utilizaron 22 funciones de prueba que se muestran en la tabla 4.1. La notación de las columnas de dicha tabla es la siguiente:

Problema	Nombre del problema.
Ref.	Referencia con la descripción del problema.
n	Dimensión del problema.
mg	Número de mínimos globales.
ϵ	El criterio de finalización ($w(X) \leq \epsilon$).
L	El nivel máximo del árbol.
$Offspr(S)$	Número de nodos evaluados por el algoritmo.

La tabla 4.1 muestra las funciones divididas en tres conjuntos de prueba distintos. Se utilizó una configuración diferente para cada uno de estos conjuntos que se muestra en la tabla 4.2.

La forma del árbol de búsqueda generado depende del problema y del algoritmo de ramificación y acotación. Para ilustrar los conceptos introducidos, la figura 4.7 muestra los gráficos del número de nodos por nivel (a la izquierda) y de la secuencia-γ (a la derecha) para la función test Levy 5. Los gráficos para el resto de funciones presentadas en la tabla 4.1 se encuentran en el apéndice B. Nótese que el número de nodos por nivel y la secuencia-γ ofrecen una información similar. El gráfico correspondiente a la secuencia-γ tiene una linea punteada en color rojo en $\gamma = 0.5$ para diferenciar entre $\gamma_i < 0.5$, que significa un incremento en el número de nodos del nivel i al nivel $i+1$, $\gamma_i > 0.5$, que significa un decremento en el número de nodos y $\gamma_i = 0.5$, que significa que el número de nodos en el nivel i y en el nivel $i+1$ es el mismo. Esto se puede observar en el gráfico correspondiente al número de nodos por niveles, a la izquierda. Es interesante observar el comportamiento en zigzag de la secuencia-γ. Esto es una consecuencia de usar bisección, después de n divisiones (en concreto 2 para el problema Levy 5) el tamaño de la caja se reduce a la mitad, dando un comportamiento de rechazo diferente al de los niveles intermedios.

Para representar la predicción de *left-over* realizada por los métodos descritos en la sección 4.5, dibujaremos un gráfico para cada problema del primer conjunto de prueba. El gráfico para Levy 5 se muestra en la figura 4.8. El eje de abscisas representa el número de iteraciones del algoritmo y el eje de ordenadas el valor de *left-over* y la predicción de cada método. Solo se muestran los valores estimados cercanos al valor de *left-over* definido por la ecuación (4.15). La línea negra que

Tabla 4.1: Funciones de prueba para la predicción del *left-over*

Problema	Ref.	n	mg	ϵ	L	$Offspr(S)$
Goldstein-Price	[124]	2	1	10^{-3}	24	101 668
Levy 3	[124]	2	9	10^{-4}	36	337 786
Levy 5	[124]	2	1	10^{-5}	42	299 656
Griewank 2	[130]	2	1	10^{-9}	82	109 390
Griewank 10	[130]	10	1	10^{-6}	310	616 446
EX 1	[33]	2	1	10^{-6}	42	383 058
Branin	[124]	2	3	10^{-9}	68	146 358
Henriksen-Madsen 3	[59]	2	9	10^{-4}	36	335 896
Henriksen-Madsen 4	[59]	3	1	10^{-3}	42	216 292
Chichinadze	[37]	2	1	10^{-5}	46	697 304
Shekel 10	[124]	4	1	10^{-5}	80	8 487 156
Shekel 7	[124]	4	1	10^{-5}	80	6 939 346
Shekel 5	[124]	4	1	10^{-5}	80	313 096
Hartman 6	[124]	6	1	10^{-2}	42	877 002
Hartman 3	[124]	3	1	10^{-3}	30	454 568
Ratz 5	[124]	3	1	10^{-4}	54	2 359 306
Rosenbrock 10	[37]	10	1	10^{-5}	190	581 136
Colville	[25]	4	1	10^{-5}	84	1 211 542
Kowalik	[124]	4	1	10^{-3}	36	3 090 698
Neumaier 2	[105]	4	12	10^{-3}	48	21 399 102
Neumaier 3-10	[105]	10	1	10^{-2}	150	12 958 026
Ratz 8	[124]	9	1	10^{-4}	162	4 718 574

Tabla 4.2: Configuración del banco de pruebas para la predicción del *left-over*

Conjunto	f^* conocido	Test de monotonía	k
1	SI	NO	1 000
2	NO	NO	1 000
3	NO	SI	10 000

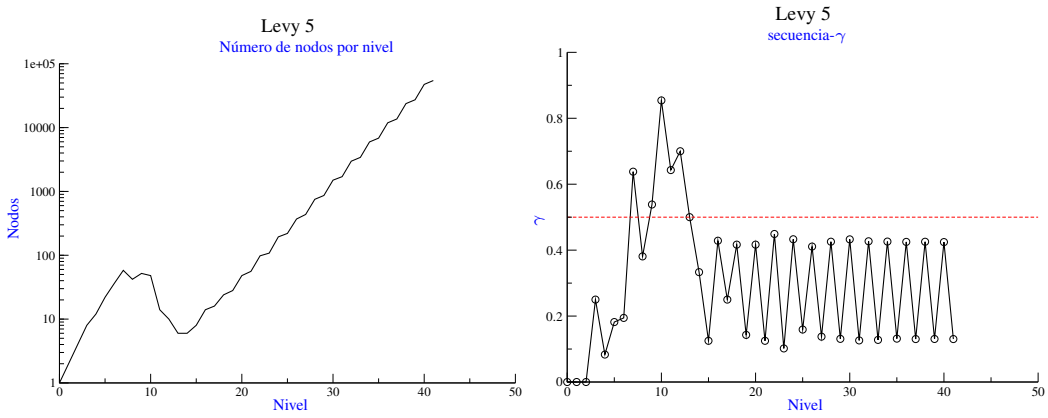

Figura 4.7: Número de nodos por nivel y secuencia-γ para el problema Levy 5 - A la izquierda se muestra el número de nodos totales evaluados en cada nivel del árbol de ramificación y acotación. A la derecha se muestra la secuencia-γ obtenida al finalizar la ejecución. El problema fue resuelto con el algoritmo 2 con una precisión de $\epsilon = 10^{-5}$.

representa el valor de *left-over* comienza en $Offspr(S)$ y disminuye en dos unidades en cada iteración, ya que dos nodos son evaluados en dicha iteración. El valor de $Offspr(S)$ para cada uno de los problemas puede encontrarse en la tabla 4.1. Por ejemplo, para el problema Levy 5 el valor de $Offspr(S)$ es de $299\,656$ nodos evaluados.

La figura 4.8 muestra como la predicción realizada por los métodos IG-LEM e IL-LEM coinciden bastante bien con el valor real del *left-over* a partir de la mitad de la ejecución, mientras que la estimación del método PL-LEM permanece bastante alejada durante todo el proceso. Además, hemos de destacar que la estimación de la profundidad de los subárboles utilizada por el método IL-LEM, consigue que las estimaciones sean buenas en momentos iniciales de la ejecución del algoritmo.

Las figuras C.1 a C.9 en el apéndice C muestran las gráficas de predicción para el resto de problemas del primer conjunto de prueba. En ellas podemos observar diferentes comportamientos en el árbol de búsqueda para los distintos problemas. Las familias de problemas como Levy 3-Levy 5, Griewank 2-Griewank 10 y Henriksen-Madsen 3-Henriksen-Madsen 4 muestran un comportamiento similar en sus árboles de búsqueda, incluso cuando los problemas en la misma familia tienen un número diferente de mínimos y/o de dimensión. Además, la mayoría de las funciones de

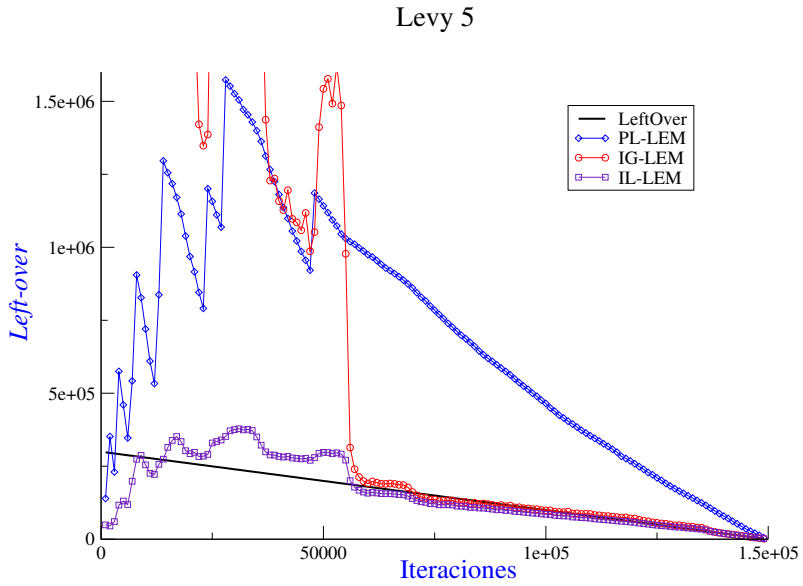

Figura 4.8: Valores de predicción del *left-over* de los distintos métodos de estimación para el problema Levy 5 - La gráfica muestra cómo los valores de predicción del *left-over* obtenidos por los métodos basados en iteraciones (IG-LEM e IL-LEM) se acercan más al valor real, comparado con el método basado en niveles PL-LEM. Las estimaciones se realizaron cada 1000 iteraciones del algoritmo 2, para una precisión $\epsilon = 10^{-5}$.

prueba muestran un número creciente de nodos en los niveles finales. Esto es debido a que el algoritmo busca todos los puntos minimizadores globales y la búsqueda que se está realizando en esos niveles está centrada en los mismos.

Los problemas Griewank 2, Griewank 10, Branin y Chichinadze proporcionan un comportamiento específico interesante: los límites inferiores para todos los nodos son igual al valor del mínimo, $\underline{F}(X) = f^* = 0,\ \forall X \in \Lambda$. Por tanto, la regla de selección *primero-el-mejor* $\underline{F}(X)$ no es relevante y debemos establecer un segundo criterio de selección. Si el segundo criterio es seleccionar el nodo con menor límite superior, $\overline{F}(X)$, entonces se produce un comportamiento similar a una búsqueda en profundidad causando que los métodos IL-LEM e IG-LEM predigan bastante mal, mientras que el estimador PL-LEM predice casi perfectamente. En cambio, si el segundo criterio es seleccionar el nodo más antiguo primero (estrategia LIFO) entonces los métodos IL-LEM e IG-LEM funcionan tan bien como el estimador PL-LEM. Por este motivo, se utiliza la estrategia LIFO como segundo criterio de

selección en todos los experimentos.

4.6.2 Resultados numéricos

Para medir la calidad de los métodos estimadores de *left-over* definimos el indicador $ARPE$ (*Average Relative Prediction Error* o media relativa del error de predicción) como:

$$ARPE = \frac{1}{N}\sum_{j=1}^{N} \frac{|LeftOver^{E}(\Lambda_j) - LeftOver(\Lambda_j)|}{LeftOver(\Lambda_j)}, \tag{4.22}$$

donde N es el número total de predicciones hechas durante el experimento. Este número varía para cada instancia y configuración de prueba. En los experimentos realizados, el indicador $ARPE$ es medido durante diferentes etapas de la ejecución del algoritmo, en concreto hemos definido los siguientes intervalos: $[0-20]\,\%, [20-40]\,\%, [40-60]\,\%, [60-80]\,\%$ y $[80-100]\,\%$. Los valores del indicador de predicción $ARPE$ se muestran en tablas separadas para cada método.

Las tablas 4.3, 4.4 y 4.5 muestran los valores de $ARPE$ para los métodos PL-LEM, IG-LEM e IL-LEM, respectivamente. Con objeto de facilitar la comparación de los tres métodos, se han ordenado de peor a mejor los problemas dentro de cada conjunto de prueba y el mejor valor global de los tres métodos se resalta en negrita.

La regla general al realizar estimaciones es que si las estimaciones actuales dan buena información sobre el futuro entonces la predicción es posible. En esta investigación esto significa que los factores de rechazo medidos ofrecen información adecuada. Resaltemos un caso extremo, la instancia del problema Rosembrock 10. La predicción realizada por el método PL-LEM es extremadamente buena durante todas las etapas de la ejecución, mientras que los otros métodos erran completamente. Profundizando un poco en los datos vemos que los valores de la secuencia-γ están entre 0 y 0.7 en todos los niveles y son bien medidos por $\hat{\gamma}$. Sin embargo, los factores de rechazo por iteraciones son medidos con valores de todo el árbol y una estimación por debajo del valor real de rechazo sobrestimará la predicción.

Si analizamos de forma más general la tabla 4.3 observamos que siete de los problemas obtienen errores de predicción por debajo de 0.5 en la mayoría de las etapas. Para estos problemas, la secuencia-γ es relativamente constante durante la

ejecución. Esto significa que las cajas en el mismo nivel tienen un factor de rechazo similar. Nótese que si $\hat{\gamma}$ predice la secuencia-γ exactamente, no se produce error de predicción, como ocurre con las funciones Griewank 2 y Griewank 10. Por otro lado, un comportamiento poco regular de la secuencia-γ conduce a grandes errores de predicción, incluso en las últimas etapas. Ejemplos de ello son las funciones Henriksen-Madsen 4, Henriksen-Madsen 3, la familia de funciones Levy, EX 1, las familias Hartman y Shekel, Colville y Kowalik.

Tabla 4.3: ARPE para el método de predicción PL-LEM

Problema	0-20 %	20-40 %	40-60 %	60-80 %	80-100 %
Henriksen-Madsen 4	**26.50**	**32.80**	21.30	13.01	6.75
Goldstein-Price	12.35	4.25	0.93	**0.34**	1.31
Henriksen-Madsen 3	**2.89**	5.90	5.32	4.53	2.97
Levy 5	2.48	4.52	4.22	3.58	3.05
Levy 3	**2.43**	4.49	4.25	3.49	3.27
EX 1	**1.64**	2.75	2.54	2.23	2.42
Branin	**1.29**	2.82	3.39	2.64	0.79
Chichinadze	**1.19**	2.24	2.25	1.02	0.76
Griewank 2	0.23	0.09	**0**	**0**	**0**
Griewank 10	**0**	**0**	**0**	**0**	**0**
Hartman 6	**874.38**	221.11	83.80	27.18	10.71
Shekel 7	**130.06**	82.96	43.23	20.16	8.04
Shekel 5	126.49	65.45	29.68	16.90	7.95
Shekel 10	**125.23**	77.04	41.66	19.63	7.97
Colville	**19.92**	**15.20**	**14.59**	12.05	6.39
Hartman 3	**4.94**	5.74	4.55	3.89	3.50
Ratz 5	**0.57**	0.42	0.23	0.37	0.06
Rosenbrock 10	**0.12**	**0.05**	**0.05**	**0.05**	**0.06**
Kowalik	**7.82**	6.87	5.98	6.32	6.82
Ratz 8	0.68	**0.37**	**0.18**	**0**	**0**
Neumaier 3-10	**0.66**	**0.49**	**0.28**	**0.24**	0.40
Neumaier 2	**0.58**	**0.24**	**0.07**	**0.14**	**0.02**

La tabla 4.4 muestra los valores $ARPE$ para el método IG-LEM. Los resultados obtenidos muestran buenas predicciones a partir de la tercera etapa (40-60 %) de la ejecución. Los resultados son buenos en la mayoría de las instancias, siendo aún mejores en la última etapa. Solo la función Colville muestra un error de predicción más alejado del resto. Podemos observar también que el error de predicción puede crecer ocasionalmente en la última etapa, esto es debido al incremento observado en el número de nodos que alcanzan el último nivel del árbol.

Tabla 4.4: ARPE para el método de predicción IG-LEM

Problem	0-20 %	20-40 %	40-60 %	60-80 %	80-100 %
Henriksen-Madsen 4	2.10E+6	8.55E+4	1.19	0.48	0.01
Goldstein-Price	40.95	**0.18**	**0.33**	1.29	0.44
Henriksen-Madsen 3	124.30	7.85	0.54	0.20	0.01
Levy 5	143.89	6.78	**0.08**	**0.05**	0.11
Levy 3	62.08	5.93	**0.07**	**0.05**	0.15
EX 1	6.49	2.43	**0.30**	**0.26**	**0.13**
Branin	3.07	**2.17**	**0.76**	**0.54**	**0.31**
Chichinadze	9.88	3.22	1.66	**0.18**	**0.25**
Griewank 2	0.09	**0**	**0**	**0**	**0**
Griewank 10	1.10E+86	0.02	0.02	0.01	**0**
Hartman 6	7.02E+4	39.45	1.44	0.19	**0.04**
Shekel 7	1.19E+5	7.08	1.41	0.57	**0**
Shekel 5	8.98E+4	11.89	0.90	0.49	0.06
Shekel 10	4.83E+4	3.89	1.31	0.60	**0**
Colville	7.55E+9	153.75	32.03	7.71	3.21
Hartman 3	21.97	**0.27**	**0.25**	**0.12**	**0.09**
Ratz 5	181.73	**0.27**	**0.22**	**0.31**	**0**
Rosenbrock 10	1.23E+27	1.99E+5	4.30E+3	28.29	0.59
Kowalik	15.57	2.35	0.35	0.27	0.39
Ratz 8	**0.66**	1.18	1.21	0.03	**0**
Neumaier 3-10	4.43E+4	4.21	0.36	0.79	0.61
Neumaier 2	2.53	0.95	0.59	0.25	0.03

La tabla 4.5 muestra el valor $ARPE$ para el método IL-LEM. En general, este método supera al resto. Obtiene las mejores predicciones en la última etapa de la ejecución (80-100 %). Más aún, se realizan buenas estimaciones desde muy pronto, a partir de la segunda etapa (20-40 %) de la ejecución del algoritmo. Si nos centramos en los valores que no están marcados en negrita, estos también son muy similares a la mejor predicción, salvo en un reducido número de casos. Las figuras en el apéndice C ilustran este comportamiento.

Los métodos IG-LEM e IL-LEM proporcionan predicciones similares en las últimas etapas, mientras que en las primeras etapas, las prediccioes que realizan ambos métodos pueden variar considerablemente.

La configuración del algoritmo se varió sistemáticamente para poner a prueba la calidad y robustez de los métodos de predicción. En primer lugar, las actualizaciones de $\overline{f^*}$ proporcionan un comportamiento impredecible a la estimación dado que no podemos saber de antemano cuando se actualizará el límite superior y como influirá esto en la regla de eliminación. En numerosos escenarios, fijamos $\overline{f^*} = f^*$ de antemano. Las diferencias mostradas en la predictibilidad no parecen ser muy grandes. El segundo aspecto tratado es incorporar el test de monotonía como regla de rechazo. La hipótesis era que esto incrementaría la predictibilidad si durante la búsqueda esta regla provee un comportamiento regular. Lo observado en los resultados es que la predictibilidad ni aumenta ni disminuye por el hecho de utilizar este test. Aparentemente, los métodos de predicción basados en el comportamiento de las últimas iteraciones son, en cierto sentido, robustas a los cambios de configuración del algoritmo. Esto resulta prometedor para la aplicación general de tales métodos.

4.7 Conclusiones

La estimación de la carga computación durante la ejecución de un algoritmo de ramificación y acotación es un gran reto. En este capítulo se han presentado tres nuevos estimadores del valor de *left-over* que pueden usarse para predecir el tiempo que resta para finalizar la ejecución del algoritmo. Todos estos métodos tienen un bajo coste computacional ya que requieren muy pocas operaciones. Están basados en la medición del factor de rechazo por niveles del árbol de búsqueda (la

Tabla 4.5: ARPE para el método de predicción IL-LEM

Problem	0-20 %	20-40 %	40-60 %	60-80 %	80-100 %
Henriksen-Madsen 4	9.45E+5	4.64E+3	**0.27**	**0.12**	**0**
Goldstein-Price	**0.84**	0.78	0.78	0.54	**0.09**
Henriksen-Madsen 3	76.43	**0.28**	**0.29**	**0.16**	**0**
Levy 5	**0.29**	**0.37**	0.15	0.12	**0.03**
Levy 3	24.20	**0.96**	0.15	0.12	**0.05**
EX 1	4.23	**1.68**	0.31	0.27	0.14
Branin	3.07	**2.17**	**0.76**	**0.54**	**0.31**
Chichinadze	7.21	2.47	**1.48**	**0.18**	**0.25**
Griewank 2	**0.04**	**0**	**0**	**0**	**0**
Griewank 10	1.10E+86	0.02	0.02	0.01	**0**
Hartman 6	2.73E+4	**7.18**	**0.35**	**0.10**	0.14
Shekel 7	489.29	**0.28**	**0.19**	**0.04**	**0**
Shekel 5	**1.04**	**0.10**	**0.08**	**0.02**	**0.01**
Shekel 10	156.18	**0.32**	**0.24**	**0.09**	**0**
Colville	5.80E+9	66.73	15.91	**3.89**	**1.07**
Hartman 3	10.55	0.31	0.30	0.17	0.11
Ratz 5	181.73	**0.27**	**0.22**	**0.31**	**0**
Rosenbrock 10	1.23E+27	1.99E+5	4.30E+3	28.29	0.59
Kowalik	7.87	**0.69**	**0.09**	**0.07**	**0.06**
Ratz 8	**0.66**	1.18	1.21	0.03	**0**
Neumaier 3-10	1.62E+3	0.57	0.67	0.60	**0.22**
Neumaier 2	2.53	0.95	0.59	0.25	0.03

llamada secuencia-γ) o por iteraciones del algoritmo. La predicción se basa en la suposición de que este factor de rechazado será similar en el futuro. Específicamente, se ha desarrollado el método PL-LEM basado en el rechazo por niveles y los métodos IG-LEM e IL-LEM basados en el rechazo por iteraciones del algoritmo. Los resultados experimentales muestran que, en general, las mejores predicciones se obtienen para los métodos basados en los factores de rechazo por iteraciones. Los resultados experimentales muestran que se pueden realizar estimaciones razonablemente buenas transcurrido el 40 % de la ejecución.

La inclusión de información adicional en los métodos mejora las predicciones pero incrementa el coste computacional. Es el caso del método IL-LEM, el cual usa una estimación de la profundidad del subárbol para cada caja y obtiene buenas estimaciones de *left-over*. Adicionalmente, esta estimación de la profundidad de los subárboles puede usarse para realizar balanceo dinámico de la carga en algoritmos paralelos. El estudio del rendimiento de tal método de balanceo se realizará en trabajos futuros. Por otro lado, el método PL-LEM merece un estudio futuro en el que se incluya información adicional para mejorar la predicción de la secuencia-γ. En el futuro, también se pretende estudiar el uso de otros indicadores de la calidad de una caja [23].

Si pude ver más allá de otros hombres, es porque me subí a hombros de gigantes.

Isaac Newton

CAPÍTULO

5

Nuevo método de acotación usando términos aditivamente separables

5.1 Introducción

En la sección 2.5 se estudió cómo se podían obtener inclusiones del rango real de una función por medio de la aritmética de intervalos. La diferencia entre el rango real de la función y la inclusión obtenida, lo que denominamos como sobrestimación del rango real o exceso de anchura (*excess width*), es de vital importancia para mejorar la eficiencia del algoritmo de optimización global intervalar. En la sección 2.5.2 se estudiaron las formas centradas, las cuales proveen un orden de convergencia cuadrático, mientras que la extensión natural a intervalos tiene un orden de convergencia lineal. En este capítulo se investigan las características de las funciones que pueden ser descompuestas en diferentes términos o subfunciones, en particular, cuando la función es suma de dos subfunciones. A este tipo de funciones las llamaremos *funciones aditivamente separables* [8, 10]. La cuestión que se investiga en este capítulo es si usando esta característica de las funciones, se pueden obtener mejores límites inferiores de la función objetivo en un intervalo. Para

responder a esta pregunta, estudiamos la separabilidad en funciones unidimensionales [12], usando la forma de Baumann y la forma LBVF (véase la sección 2.5.2, página 20) para el cálculo de los mencionados límites inferiores.

Definición 18 *La función $f : S \subset \mathbb{R} \to \mathbb{R}$ es aditivamente separable, si puede escribirse como*

$$f(x) = \sum_{j=1}^{p} f_j(x),\ x \in S. \tag{5.1}$$

Sabemos que

$$\min_{x \in S} f(x) \geq \sum_{j=1}^{p} \min_{S} f_j(x). \tag{5.2}$$

Sea $\underline{F}_j(S)$ un límite inferior de f_j sobre S. Entonces tenemos que

$$\min_{x \in S} f(x) \geq \sum_{j=1}^{p} \underline{F}_j(S). \tag{5.3}$$

En la sección 2.5 estudiamos cómo computar un límite inferior de la función f en el intervalo $X \subseteq S$, denotado por $\underline{F}(X)$. Si consideramos funciones que tienen una estructura aditivamente separable (5.1), la cuestión que estudiamos en este capítulo es para qué casos se cumple:

$$\underline{F}(X) \leq \sum_{j=1}^{p} \underline{F}_j(X), \tag{5.4}$$

es decir, la suma de los límites inferiores de las subfunciones f_j en el intervalo X es mejor que el límite inferior calculado para la función f en el mismo intervalo X. Alternativamente, también queremos encontrar formas de combinar minorantes de los términos separables para obtener mejores límites inferiores.

Ejemplo 9 *Consideremos la función $f(x) = f_1(x) + f_2(x) = (x+1)^2 + (x-1)^2$ en el intervalo $X = [-2, 2]$. El mínimo de las subfunciones f_1 y f_2 es 0, mientras que el mínimo de la función f es 2, en el punto $x^* = 0$. Si usamos la forma de Baumann para calcular los límites inferiores en el intervalo X, el límite inferior para $f(x \in X)$ es $\underline{F}(X) = -14$, mientras que el límite inferior para las subfunciones es $\underline{F}_j(X) = -6$, de modo que $\underline{F}_1(X) + \underline{F}_2(X) = -12$. Lo que significa que aprovechando la característica de separabilidad de la función f se ha obtenido un mejor límite inferior, esto es, $\underline{F}_1(X) + \underline{F}_2(X) > \underline{F}(X)$, ilustrando la ecuación (5.4).*

La figura 5.1 ilustra el ejemplo 9 y muestra gráficamente los puntos de Baumann (véase la sección 2.5.2) y los minorantes obtenidos. Nótese que el punto b^- coincide con el punto mínimizador x^* para funciones cuadráticas. La figura 5.1 muestra la posibilidad de tener diferentes puntos b^- para cada subfunción y su uso para la obtención de limites inferiores para cada subfunción. La suma de los límites inferiores obtenidos para las subfunciones permite, para este caso, obtener un limite inferior mejor que el obtenido con el punto b^- de la función compuesta.

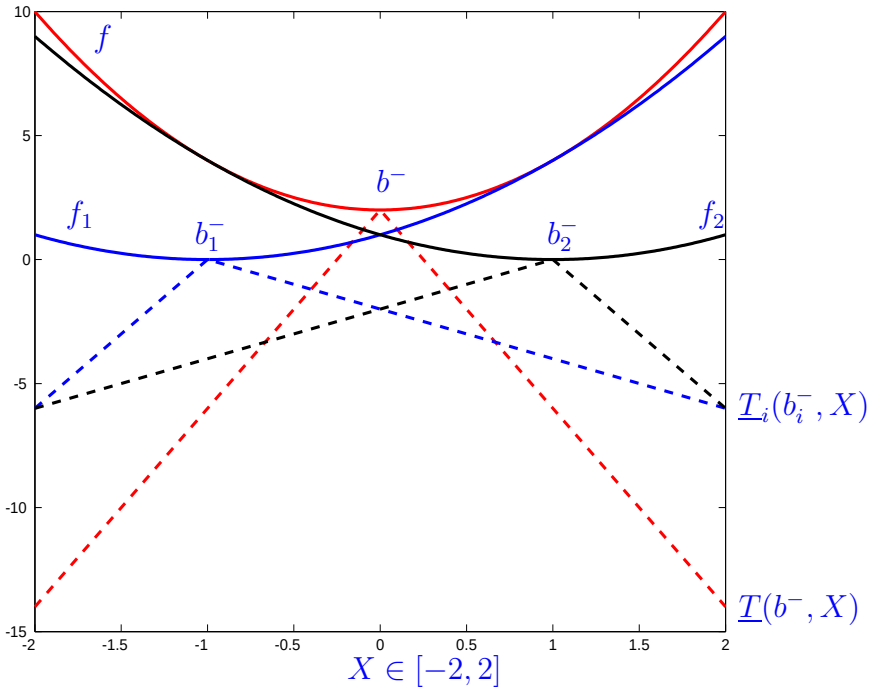

Figura 5.1: Ilustración de la forma de Baumann aditivamente separable - Evaluando la función $f(x) = f_1(x) + f_2(x) = (x+1)^2 + (x-1)^2$ en el intervalo $X = [-2, 2]$ con la forma de Baumann obtenemos un límite inferior $\underline{T}(b^-, X) = -14$, mientras que utilizando la forma de Baumann separable obtenemos un mejor límite inferior $\underline{T_1}(b_1^-, X) + \underline{T_2}(b_2^-, X) = -12$.

5.2 Acotación usando términos aditivamente separables

Consideraremos el caso en el que una función unidimensional es aditivamente separable en dos subfunciones, es decir:

$$f(x) = f_1(x) + f_2(x).$$

Si ambas subfunciones son monótonas en la misma dirección (bien $\underline{F}'_j(X) > 0$, o bien $\overline{F}'_j(X) < 0,\ \forall j$), entonces la función f es monótona en X. Pero si ambas subfunciones son monótonas en direcciones opuestas, puede ocurrir que la función global f no sea monótona, es decir, $0 \in F'(X)$. Más aún, una de las subfunciones puede ser monótona y la otra no. En general, cuando la función completa es monótona, esto es, $0 \notin F'(X)$, el valor de la función en uno de los puntos extremos es el mínimo en el intervalo X. Por tanto, el límite inferior se obtiene como el $\min\{\underline{F}(\underline{X}), \underline{F}(\overline{X})\}$. Por tanto, usar términos separables para calcular el límite inferior de f en un intervalo X podría mejorarlo solo en el caso de que $0 \in F'(X)$, es decir, la función no sea monótona en X.

A continuación se estudian las características de la forma de Baumann y LBVF usando términos separables y se estudia un nuevo minorante con el que se pueden obtener mejores límites inferiores que los obtenidos con LBVF.

5.2.1 Baumann con términos aditivamente separable (ASB)

La forma de Baumann aditivamente separable, denotada por $ASB(X)$, se puede construir de forma directa evaluando la expresión (2.26) para las dos subfunciones en sus respectivos puntos de Baumann y sumando los resultados,

$$f(X) \geq ASB(X) = \underline{T}_1(b_1^-, X) + \underline{T}_2(b_2^-, X). \tag{5.5}$$

La cuestión es si la ecuación (5.5) obtiene siempre mejores límites inferiores que la forma de Baumann estándar (2.27). Esta cuestión la investigaremos experimentalmente en la sección 5.3.

5.2.2 LBVF con términos aditivamente separables (ASLBVF)

Para obtener la forma LBVF separable, seguimos el mismo razonamiento realizado para la forma de Baumann separable, es decir, mediante la evaluación de la ecuación (2.32) para cada una de las dos subfunciones:

$$f(X) \geq ASLBVF(X) = \underline{\varphi m}_1(X) + \underline{\varphi m}_2(X),\ 0 \in F_1'(X),\ 0 \in F_2'(X). \quad (5.6)$$

Sin embargo, esta ecuación solo permite obtener límites inferiores válidos si ambas subfunciones no son monótonas en el intervalo evaluado. La cuestión es si de esta forma se obtienen mejores límites inferiores a los obtenidos usando (2.38). Como en el caso anterior, lo investigaremos experimentalmente en la sección 5.3.

5.2.3 Límite inferior basado en un nuevo minorante φ: $ASLB_\varphi$

Nos centraremos en los minorantes que se obtienen con la forma LBVF en las subfunciones para tratar de obtener mejores límites inferiores que los obtenidos con $\underline{\varphi m}(X)$ (véase la ecuación (2.32), página 23), sin preocuparnos de la monotonía o no de las subfunciones. Nótese otra vez que estamos interesados en el caso en el que la función compuesta no es monótona. Consideremos la suma de los dos términos minorantes separados

$$\varphi(x) = \varphi m_1(x) + \varphi m_2(x),$$

donde φm_i se define por (2.30). En primer lugar, nótese que φ es una función minorante lineal definida a trozos o por partes, siendo el máximo de cuatro términos afines diferentes:

$$\varphi(x) = \text{máx} \left\{ \begin{array}{l} \varphi l(x) := \varphi l_1(x) + \varphi l_2(x) \\ \varphi a(x) := \varphi l_1(x) + \varphi r_2(x) \\ \varphi b(x) := \varphi r_1(x) + \varphi l_2(x) \\ \varphi r(x) := \varphi r_1(x) + \varphi r_2(x) \end{array} \right\}. \quad (5.7)$$

Por tanto, se puede observar que $\varphi(x)$ es un mejor minorante que $\varphi m(x)$.

Teorema 1 *Sea* $\forall x \in X, f(x) = f_1(x) + f_2(x)$ *y* φl, φr *y* φm *definidos por (2.28), (2.29) y (2.30).* $\forall x \in X, \varphi m_1(x) + \varphi m_2(x) \geq \varphi m(x)$.

Demostración.
Dada la equivalencia (5.7), tenemos que

$$\varphi m(x) = \max\{\varphi l(x), \varphi r(x)\} \leq \max\{\varphi l(x), \varphi a(x), \varphi b(x), \varphi r(x)\} = \varphi(x).$$

■

Dado el teorema anterior, estamos interesados en el valor mínimo de este minorante, es decir, $\min_{x \in X} \varphi(x)$. Para encontrar este mínimo, consideramos varias propiedades de la función minorante lineal por partes $\varphi(x)$. El máximo de $\varphi(x)$ se puede encontrar típicamente en los bordes.

Proposición 1 *Sea* $0 \in F'(X)$ *y* φ *definida por (5.7). Entonces*

$$\varphi(\underline{X}) = \varphi l(\underline{X}) \geq \max\{\varphi a(\underline{X}), \varphi b(\underline{X}), \varphi r(\underline{X})\}$$

y

$$\varphi(\overline{X}) = \varphi r(\overline{X}) \geq \max\{\varphi a(\overline{X}), \varphi b(\overline{X}), \varphi l(\overline{X})\}.$$

Demostración.
Las desigualdades se obtienen tras escribir explícitamente los términos en (5.7) y considerando $\varphi l_j(\underline{X}) \geq \varphi r_j(\underline{X})$ y $\varphi r_j(\overline{X}) \geq \varphi l_j(\overline{X})$. ■

Proposición 2 *Sea* $0 \in F'(X)$ *y* φ *definida por (5.7). La función* φ *es un minorante convexo de* f *sobre* X.

Demostración.
Esto se demuestra directamente a partir del hecho de que φ es el máximo de varias funciones convexas, según (5.7). ■

Puesto que φ es una función lineal por partes, solo tiene un mínimo, bien con un único punto minimizador o infinitos puntos minimizadores. Las condiciones de primer orden para un mínimo de una función líneal por partes establecen que debe haber un subgradiente 0. Los puntos con más de una derivada (subgradiente) son los puntos de intersección entre los minorantes afines. Escribiendo todos los términos se demuestra que estos puntos de intersección son típicamente y_1 y/o y_2 (2.31):

- $\varphi l(x) = \varphi a(x) \rightarrow \varphi l_2(x) = \varphi r_2(x)$ con solución y_2,

- $\varphi l(x) = \varphi b(x) \rightarrow \varphi l_1(x) = \varphi r_1(x)$ con solución y_1,
- $\varphi r(x) = \varphi a(x) \rightarrow \varphi r_1(x) = \varphi l_1(x)$ con solución y_1,
- $\varphi r(x) = \varphi b(x) \rightarrow \varphi r_2(x) = \varphi l_2(x)$ con solución y_2,
- $\varphi l(x) = \varphi r(x)$ con solución y (2.31), siendo el punto mínimo de φm, no es un punto mínimo único de φ debido al teorema 1; $\varphi(x) \geq \varphi m(x)$ es un mejor minorante.
- $\varphi a(x) = \varphi b(x)$ se puede mostrar que obtiene el siguiente punto de intersección

$$\hat{x} = \frac{\underline{X}(\underline{F'}_1 - \underline{F'}_2) + \overline{X}(\overline{F'}_2 - \overline{F'}_1) + \underline{F}_1(\overline{X}) - \underline{F}_1(\underline{X}) + \underline{F}_2(\underline{X}) - \underline{F}_2(\overline{X})}{w(F'_2) - w(F'_1)}. \tag{5.8}$$

  Probaremos que $\hat{x}$ no es un candidato para el punto mínimo de φ.

La siguiente proposición muestra que podemos centrarnos en el intervalo entre los puntos y_1 e y_2.

Proposición 3 *Sea* $0 \in F'(X)$, φ *definida por (5.7) e* y_j *por (2.31). Entonces* $\min_{x \in X} \varphi(x)$ *se encuentra en el intervalo* $[\min\{y_1, y_2\}, \max\{y_1, y_2\}]$.

Demostración.
Siguiendo la argumentación de la proposición 1 y habiendo descartado y como punto mínimo único, tenemos

- $\forall x \in [\underline{X}, y_2]\ \varphi l(x) \geq \varphi a(x)$
- $\forall x \in [\underline{X}, y_1]\ \varphi l(x) \geq \varphi b(x)$

y la derivada $\varphi l' \leq 0$. Así, no hay un punto mínimo único en el intervalo $[\underline{X}, \min\{y_1, y_2\})$. Por simetría con respecto a φr se puede obtener que no hay un punto mínimo único en el intervalo $(\max\{y_1, y_2\}, \overline{X}]$ ■

Ahora, podemos caracterizar el punto mínimo exacto de $\varphi(x)$ sobre X, que ayudará a generar un mejor límite inferior que el minorante φm.

Teorema 2 *Sea* $0 \in F'(X)$, φ *definido por (5.7) e* y_j *por (2.31). Entonces* $\min_{x \in X} \varphi(x)$ *se encuentra con seguridad en* $\{y_1, y_2\}$.

Demostración.

La proposición 3 limita el mínimo al intervalo $[\text{mín}\{y_1, y_2\}, \text{máx}\{y_1, y_2\}]$. Consideremos $y_1 \leq y_2$.

En el intervalo $[y_1, y_2]$ tenemos que $\varphi r_1(x) \geq \varphi l_1(x)$ y $\varphi l_2(x) \geq \varphi r_2(x)$, tal que $\varphi(x) = \varphi b(x) \geq \varphi a(x)$. Por tanto, $\varphi(x)$ es afín en $[y_1, y_2]$, encontrándose su mínimo en uno de los extremos del intervalo.

Consideremos el caso $y_2 \leq y_1$.

En el intervalo $[y_2, y_1]$ tenemos que $\varphi l_1(x) \geq \varphi r_1(x)$ y $\varphi r_2(x) \geq \varphi l_2(x)$, tal que $\varphi(x) = \varphi a(x) \geq \varphi b(x)$. Por tanto, $\varphi(x)$ es afín en $[y_2, y_1]$, encontrándose su mínimo en uno de los extremos del intervalo.

■

Nótese que el razonamiento de la demostración también muestra que el punto $\hat{x}$, donde φa y φb intersectan, no puede ser un punto mínimo único. La teoría mostrada hasta ahora nos provee una nueva función minorante para obtener un límite inferior, denotada por $ASLB\varphi$ y definida por

$$f(X) \geq ASLB\varphi(X) = \underline{\varphi}(X) = \text{mín}\{\varphi(y_1), \varphi(y_2)\}. \tag{5.9}$$

5.3 Evaluación experimental de las funciones de inclusión separables

En primer lugar discutiremos las características de los experimentos diseñados y después mostraremos y discutiremos los resultados numéricos obtenidos.

5.3.1 Diseño de los experimentos

Para medir la efectividad de las funciones de inclusión separables, usaremos el algoritmo 2 sobre 18 funciones de prueba unidimensionales. La medición de los límites inferiores separables y la comparación con sus respectivas versiones no separables se realiza sobre aquellas cajas que no han sido rechazadas por el algoritmo 2, tras la ejecución del test de rango, test de corte y test de monotonía. El criterio de finalización utilizado es $\epsilon = 10^{-6}$ para todas las funciones test.

La tabla 5.1 describe las funciones de prueba empleadas. La primera columna indica el índice de la función, f_1 y f_2 son los términos aditivos de la función $f = f_1 + f_2$ y la última columna proporciona una referencia a la literatura donde se describe la función. El espacio de búsqueda para todas las funciones está establecido a $S = [0.5, 20]$, salvo las instancias N. 6 y N. 11 donde $S = [-10, 10]$.

Tabla 5.1: Funciones de prueba aditivamente separables

N.	f_1	f_2	Ref.
1	e^{-3x}	$-\sin^3 x$	[123]
2	xe^{-x^2}	$\sin xe^{-x^2}$	[123]
3	$\sin x + \ln x$	$\sin \frac{10x}{3} - 0.84x$	[123]
4	$x^4 - 10x^3$	$35x^2 - 50x + 24$	[123]
5	$24x^4 - 142x^3$	$303x^2 - 276x + 93$	[123]
6	$2x^2$	$-\frac{3}{100}e-(200(x - 0.0675))^2$	[123]
7	$\frac{x^2}{20}$	$-\cos x + 2$	[123]
8	x^2	$-\cos(18x)$	[123]
9	$x^4 - 12x^3$	$47x^2 - 60x - 20e^{-x}$	[123]
10	$x^6 + 250$	$-15x^4 + 27x^2$	[123]
11	$\sin^2(1 + \frac{x-1}{4})$	$(\frac{x-1}{4})^2$	[123]
12	$(x - x^2)^2$	$(x - 1)^2$	[123]
13	$-\sum_{k=1}^{5} k \sin[(k + 1)x + k] + 3$	$(3x - 1.4)\sin(18x) + 1.7$	[22]
14	$-x + \sin(3x) + 1$	$(3x - 1.4)\sin(18x) + 1.7$	[22]
15	$-\sum_{k=1}^{5} k \sin[(k + 1)x + k] + 3$	$\sum_{k=0}^{5} k \cos[(k + 1)x + k] + 12$	[22]
16	$-\sum_{k=1}^{5} k \sin[(k + 1)x + k] + 3$	$\cos x - \sin 5x + 1$	[22]
17	$-x + \sin(3x) + 1$	$\sum_{k=0}^{5} k \cos[(k + 1)x + k] + 12$	[22]
18	$-x + \sin(3x) + 1$	$\sum_{k=1}^{5} -cos[(k + 1)x] + 4$	[22]

5.3.2 Resultados numéricos

Para medir la efectividad de los métodos separables se aplicarán los nuevos métodos de calculo de límites inferiores solo a aquellos intervalos que no hayan sido eliminados por los test de rechazo aplicados en el algoritmo 2, recordemos que los

test que se aplican son el test de rango, el test de corte y el test de monotonía. Concretamente, evaluamos las siguientes funciones de inclusión para obtener límites inferiores:

- Baumann calculado como $\underline{T_1}(b^-, X)$, véase (2.27).
- Baumann aditivamente separable ASB, véase (5.5).
- La forma $LBVF$ calculado como $\underline{\varphi m}(X)$, véase (2.32).
- La forma $LBVF$ aditivamente separable $ASLBVF$, véase (5.6).
- $ASLB\varphi(X)$, véase (5.9).

La tabla 5.2 muestra el número de cajas que se rechazarían de forma adicional para cada una de las funciones de inclusión anteriores. Las columnas de esta tabla muestran, de izquierda a derecha: el número de la función de prueba, número de iteraciones del algoritmo, número de intervalos rechazados por $\underline{F}(X) > \overline{f^*}$ (test de rango y test de corte, RT), número de intervalos rechazados por el test de monotonía (MT) y número de intervalos que podrían haber sido rechazados usando otros tipos de cálculos del límite inferior. $ASLBVF$ se ejecuta solo si ambas subfunciones son no monótonas, mientras que $ASLB\varphi(X)$ se usa independientemente de que las subfunciones sean monótonas o no.

Los métodos basados en el LBVF muestran el mejor nivel de rechazo adicional para este conjunto de funciones de prueba. Además, usando separabilidad, solo para la función número 2, $ASLB\varphi(X)$ podría rechazar solo un intervalo adicional más. Un dato interesante es que para la forma de Baumann, usar separabilidad no parece una buena idea ya que el rechazo adicional es peor que la forma de Baumann no separable.

A continuación, nos centraremos en la mejora de los límites inferiores desde la perspectiva separable. Para hacer esto, compararemos las formas de Baumann y LBVF con sus variantes separables, junto con el límite inferior obtenido por el nuevo minorante $ASLB\varphi$.

La tabla 5.3 muestra el número de intervalos donde el límite inferior basado en la separabilidad de la función es mejor (+) y peor (-) comparado con su respectiva versión no separable. Como era de esperar a partir de los resultados mostrados en la

Tabla 5.2: Poder de rechazo adicional de las funciones de acotación

		Rechazado por		Rechazos adicionales posibles				
N.	Iter	RT	MT	$Baumann$	ASB	$LBVF$	$ASLBVF$	$ASLB\varphi$
1	25	16	9	0	0	0	0	0
2	114	110	5	0	0	1	1	2
3	27	5	22	0	0	0	0	0
4	1272	118	1093	878	57	1006	1006	1006
5	1529	313	1153	1244	156	1340	1340	1340
6	25	13	12	0	0	0	0	0
7	27	4	23	0	0	0	0	0
8	25	8	17	0	0	0	0	0
9	202	82	117	94	33	121	121	121
10	105	7	94	15	0	38	38	38
11	34	5	29	0	0	0	0	0
12	26	21	4	0	0	0	0	0
13	32	15	17	0	0	0	0	0
14	25	12	13	0	0	0	0	0
15	146	74	69	8	2	17	17	17
16	129	67	60	2	2	7	7	7
17	39	20	18	1	1	1	1	1
18	27	8	19	0	0	0	0	0

tabla 5.2, considerar Baumann desde una perspectiva separable, ASB, ofrece peores límites inferiores. De manera similar, la forma $ASLBVF$ tampoco ofrece una ganancia significativa, y en numerosas ocasiones los límites inferiores calculados empeoran la versión no separable. Sin embargo, empleando la función $ASLB\varphi$, observamos mejoras para todas las funciones de prueba en un gran número de intervalos. Además, los límites inferiores calculados con el nuevo minorante lineal son siempre, al menos, tan buenos como la versión no separable LBVF.

Tabla 5.3: Mejora del límite inferior de las funciones de acotación separables

	ASB		$ASLBVF$		$ASLB\varphi$	
N.	+	-	+	-	+	-
1	24	0	0	0	1	0
2	20	0	11	0	19	0
3	10	19	1	4	20	0
4	9	1324	2	2	782	0
5	10	1582	0	0	1102	0
6	16	10	1	7	17	0
7	6	21	0	0	16	0
8	17	14	0	0	22	0
9	10	197	3	1	174	0
10	9	100	0	0	79	0
11	6	28	1	1	20	0
12	34	2	1	14	21	0
13	25	11	6	5	29	0
14	17	14	2	7	23	0
15	76	73	19	41	127	0
16	69	64	20	37	111	0
17	24	21	7	9	34	0
18	14	16	3	7	21	0

5.4 Conclusiones

La cuestión que nos planteamos al comienzo de este capítulo era si el uso de la separabilidad aditiva de una función puede mejorar los límites inferiores, basándonos en los límites de las subfunciones. Se han presentado las versiones separables para las formas de Baumann y LBVF. La versión separable de la forma de Baumann ha demostrado ser peor que la original, mientras que la versión separable de la forma LBVF no ofrece una ganancia significativa. Además, hemos demostrado que la variante $ASLB\varphi$, basada en un nuevo minorante afín, siempre obtiene mejores o iguales límites inferiores que la versión estandar LBVF. Sin embargo, las mejoras en los límites inferiores proporcionados por el minorante $ASLB\varphi$ no conducen

a un incremento en la capacidad de rechazo de la versión estándar LBVF para el conjunto de funciones de prueba, si bien es cierto que el margen de mejora para funciones de una dimensión es menor que para funciones con más dimensiones. En investigaciones futuras nos centraremos en la cuestión de cómo extender $ASLB\varphi$ para funciones de más de una dimensión. Otra cuestión interesante que se puede abordar en un trabajo posterior es la de diseñar métodos específicos para términos multiplicativamente separables.

Estamos todos de acuerdo en que tu teoría es loca. La cuestión que nos divide es si es lo suficientemente loca para tener una posibilidad de ser correcta.

Niels Borg

CAPÍTULO

6

Optimización global intervalar de funciones aditivamente separables con variables comunes

6.1 Introducción

La complejidad de resolver el problema (1.2) crece exponencialmente con la dimensión. En muchos casos, la función objetivo puede descomponerse en varias subfunciones de dimensiones menores, como por ejemplo en [93]. Si esto ocurre de forma aditiva y las variables se pueden dividir en grupos que no se solapan, la llamaremos *función aditiva completamente separable*. Lo habitual, en cambio, es que las subfunciones compartan alguna variable común, a este tipo de funciones las llamaremos *función aditivamente separable con variables comunes*. Además, es posible que haya más de una variable común y que esta o estas no sean compartidas por todas las subfunciones. En este capítulo estudiaremos de forma teórica las funciones n-dimensionales que pueden ser separadas en dos o más subfunciones, compartiendo todas ellas una única variable común [8, 11]. La cuestión que nos

planteamos es si el diseño de métodos especificos para resolver este tipo de problemas permite disminuir la complejidad del problema con respecto a la perspectiva no separable. La implementación práctica y posterior estudio numérico se realiza para funciones separables que pueden descomponerse en dos subfunciones con una única variable común.

A continuación resumimos la notación usada en este capítulo. El espacio de búsqueda es $S = (S_1, \ldots, S_n) \in \mathbb{I}^n$ y el conjunto de índices de variables es $I = \{1, \ldots, n\}$. El conjunto de índices $I^{[j]} = \{i_1, \ldots, i_{n^{[j]}}\} \subseteq I$ con $n^{[j]} = |I^{[j]}|$ elementos, se usa para denotar subgrupos de variables y sus correspondientes regiones de búsqueda $S^{[j]} \in \mathbb{I}^{n^{[j]}}$. A continuación definimos más formalmente los conceptos de *función aditivamente separable* y *función aditiva completamente separable*.

Definición 19 *La función $f : S \subset \mathbb{R}^n \to \mathbb{R}$ es aditivamente separable, si $\exists p > 1$ tal que f puede escribirse como*

$$f(x) = \sum_{j=1}^{p} f^{[j]}(x^{[j]}),\ x^{[j]} \in S^{[j]}. \tag{6.1}$$

Definición 20 *La función $f : S \subset \mathbb{R}^n \to \mathbb{R}$ es completamente separable, si puede escribirse como (6.1) e $I^{[j]} \cap I^{[k]} = \emptyset,\ \forall k, j \in \{1, .., p\}, j \neq k$.*

El problema (1.2) puede resolverse considerando la adición de subproblemas para funciones completamente separables:

$$\min_{x \in S} f(x) = \sum_{j=1}^{p} \min_{x^{[j]} \in S^{[j]}} f^{[j]}(x^{[j]}). \tag{6.2}$$

En general, el problema (6.2) es más fácil de resolver que (1.2), con menor dimensión de los subproblemas. Sin embargo, los subproblemas con una estructura aditiva suelen tener variables comunes que son compartidas por las subfunciones (véase [93]). Nocedal y Wright mencionan en su libro [107] el concepto de problema parcialmente separable obteniendo ventajas numéricas en métodos Quasi-Newton cuando la matriz Hessiana es dispersa.

En este capítulo nos centraremos en el caso en el que solo una variable común aparece en todas las subfunciones, aunque la teoría que presentaremos es válida y

aplicable para más de una variable común. Sin pérdida de generalidad, la variable común será considerada la última, y la denotaremos por x_n. Como hemos dicho, la variable común aparece en todas las subfunciones, $n \in I^{[1]} \cap \ldots \cap I^{[p]}$. Además, los resultados numéricos se restringen para el caso $p = 2$.

Para separar la variable común x_n del resto de variables no comunes, las últimas las denotaremos por $z^{[j]} \in T^{[j]}$ tal que $S^{[j]} = (T^{[j]}, S_n)$. De tal forma que el problema a investigar puede escribirse como

$$\min_{z^{[j]} \in T^{[j]}, x_n \in S_n} \left\{ f(x) = \sum_{j=1}^{p} f^{[j]}(z^{[j]}, x_n) \right\}, \tag{6.3}$$

donde cada vector $x^{[j]} = (z^{[j]}, x_n) \in \mathbb{R}^{n^{[j]}}$ debe cumplir las restricciones individuales $x^{[j]} \in S^{[j]} \in \mathbb{I}^{n^{[j]}}$ y la variable común $x_n \in S_n \in \mathbb{I}$. Hay varias maneras de denotar a la variable común dependiendo de su contexto. Denotaremos la variable común con x_n cuando $x \in S$ y con $x_{n^{[j]}}$ cuando $x \in S^{[j]}$.

Ejemplo 10 *Consideremos la función* $f(x) = x_1^2 + x_1 x_3 - 2x_2 x_3 + x_2^2 + \frac{1}{2}x_3^2 + x_3$. *Podemos establecer* $z^{[1]} = x_1$ *y* $z^{[2]} = x_2$, *tal que* $x^{[1]} = (z^{[1]}, x_3)$ *y* $x^{[2]} = (z^{[2]}, x_3)$. *De esta forma, la función* f *puede escribirse como*

$$f(x) = f^{[1]}(x^{[1]}) + f^{[2]}(x^{[2]}),$$

donde

$$f^{[1]}(z^{[1]}, x_3) = x_1^2 + x_1 x_3 + \frac{1}{2}x_3^2 + x_3 = (z_1^{[1]})^2 + z_1^{[1]} x_3 + \frac{1}{2}x_3^2 + x_3$$

y

$$f^{[2]}(z^{[2]}, x_3) = x_2^2 - 2x_2 x_3 = (z_1^{[2]})^2 - 2z_1^{[2]} x_3.$$

6.2 Resolución de problemas aditivamente separables mediante la descomposición en subfunciones

La cuestión es cómo hacer uso de las características de una función separable con variables comunes para construir métodos y técnicas específicas para un algoritmo de optimización global intervalar que resuelva el problema (6.3), y aproveche que

las subfunciones tienen menor dimensión ($n^{[j]}$) que la función compuesta (n). Describiremos y desarrollaremos una posible perspectiva, aunque pueden existir más. La perspectiva propuesta considera copias de la variable común $X_n^{[j]}$, $X_n \subseteq S_n$ para cada subfunción. En un algoritmo de ramificación y acotación esto significa mantener listas $\Lambda^{[j]}$ y $\Omega^{[j]}$ para cada subfunción $f^{[j]}$ en un espacio $n^{[j]}$-dimensional. Esta perspectiva la denominaremos DS (Descomposición en Subfunciones).

Ejemplo 11 *Consideremos el ejemplo 10 desde la perspectiva DS. La función f tiene un mínimo global de -85 en $x = (5, -10, -10)$. En cada subfunción tendremos copias $x_3^{[1]}, x_3^{[2]}$ de la variable x_3. Relajando la restricción $x_3^{[1]} = x_3^{[2]}$ podemos calcular un límite inferior de f sobre $[-10, 10]^3$ minimizando $f^{[1]}$ y $f^{[2]}$. Minimizando $f^{[1]}(z^{[1]}, x_3^{[1]})$ da un único punto mínimo $(z^{[1]}, x_3^{[1]}) = (1, -2)$ con el valor de la función $f^{[1]}(z^{[1]}, x_3^{[1]}) = -1$. De forma similar, $f^{[2]}(z^{[2]}, x_3^{[2]})$ tiene un valor mínimo de -100 en los vértices $(-10, -10)$ y $(10, 10)$, tal que el límite inferior global resultante es -101, que es un límite inferior válido para f.*

En lugar de disponer de una única lista Λ como en el algoritmo 2, con la perspectiva DS tendríamos un conjunto de p listas $\Lambda^{[j]}$ para almacenar los subproblemas de dimensión $n^{[j]}$ para cada subfunción $f^{[j]}$. En cada una de las listas se almacenan las subcajas pertenecientes a cada subfunción, $\Lambda^{[j]} = \{X = (Z^{[j]}, X_{n^{[j]}})\}$. Denotaremos el límite inferior de una subcaja como $\underline{F}^{[j]}(X)$. El algoritmo finaliza cuando todas las listas $\Lambda^{[j]}$ están vacías. Las cajas finales se almacenan en $\Omega^{[j]}$. Para cada subfunción j, la cota inferior de la subfunción en el correspondiente subespacio de búsqueda es el mínimo límite inferior de las subcajas almacenadas en $\Lambda^{[j]} \cup \Omega^{[j]}$:

$$lb^{[j]} = \min_{X \in \Lambda^{[j]} \cup \Omega^{[j]}} \underline{F}^{[j]}(X). \tag{6.4}$$

Nótese que al terminar la ejecución del algoritmo, las listas finales $\Omega^{[j]}$ tienen que combinarse para obtener la lista de cajas finales Ω en el espacio n-dimensional. Esto requiere una última fase en el que se combinen aquellas cajas que tienen el mismo rango en la variable común.

La perspectiva DS puede ofrecer diversas ventajas con respecto al algoritmo estándar (véase el algoritmo 2) para problemas de grandes dimensiones. En primer lugar, el número de cajas generadas por el algoritmo 2, en el peor caso, se comporta de forma exponencial a medida que el número de dimensiones de la función

objetivo aumenta. En segundo lugar, para problemas de muchas dimensiones la sobrestimación de la aritmética de intervalos puede ser mucho mayor que en problemas de menor dimensión cuando existen múltiples ocurrencias de las variables. Si la razón entre el número de variables no comunes y variables comunes es alto, usar la descomposición puede suponer la obtención de ciertas ventajas.

6.3 Propiedades de las funciones separables con una variable común desde la perspectiva DS

Estas propiedades son de interés para evitar que la búsqueda del mínimo se centre en áreas definidas para las subfunciones donde no se encuentra la solución global y para obtener cotas de la función objetivo lo más ajustadas posibles.

6.3.1 Acotación

Debemos distinguir los límites inferiores de las subfunciones $f^{[j]}$ del límite inferior de la función compuesta f. El límite inferior de la subfunción es relativamente fácil de calcular usando funciones de inclusión $F^{[j]}(X \subseteq S^{[j]})$ (véase la sección 2.5).

Propiedad 5 *Por la propiedad 3 (página 20),* $lb^{[j]}$ (6.4) *es un límite inferior de* $f^{[j]}$ *sobre* $S^{[j]}$.

Existen diversas maneras de obtener un límite inferior de f. A continuación mostramos diferentes aproximaciones, siguiendo un incremento en su complejidad y efectividad.

Propiedad 6 *Por la definición 19 (página 80),* $\sum_{j=1}^{p} lb^{[j]}$ *es un límite inferior de* f *sobre* S.

La función f está definida en un espacio n-dimensional por lo que una inclusión de f en $X \subseteq S^{[j]}$ no está definida. Como alternativa, se pueden buscar los menores valores que la función f puede alcanzar cuando el intervalo X contiene los subespacios de dimensión $n^{[j]}$ con mejores soluciones para cada subfuncion, pero compartiendo valores de la variable común. Para ello, es conveniente definir para

cada intervalo $Y \subset S_n$ en el espacio de la variable común y para cada subfunción j

$$\Psi^{[j]}(Y) = \arg\min\{\underline{F}^{[j]}(X) \mid X \in \Lambda^{[j]} \cup \Omega^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\}, \tag{6.5}$$

como la subcaja X en $\Lambda^{[j]} \cup \Omega^{[j]}$ con el menor valor del límite inferior de $f^{[j]}$ y cuyo rango en la variable común está solapado con Y.

Proposición 4 *Dado un intervalo Y de S_n obtenido por una división iterativa uniforme, $\underline{F}^{[j]}(\Psi^{[j]}(Y) = (Z, X_{n^{[j]}}))$ es un límite inferior de $f^{[j]}$ sobre $(T^{[j]}, Y)$.*

Demostración. Para obtener un límite inferior de $f^{[j]}$ sobre $(T^{[j]}, Y)$, nos centramos en $\Psi^{[j]}(Y)$ que tiene el menor $\underline{F}^{[j]}(Z, X_{n^{[j]}})$ con $X_{n^{[j]}} \cap Y \neq \emptyset$. Consideremos todos los casos:

- $Y \subseteq X_{n^{[j]}}$, el resultado sigue de la definición de $\Psi^{[j]}(Y)$ y la propiedad de isotonía de las funciones de inclusión de la aritmética de intervalos (véase la propiedad 1 en la página 19).

- $X_{n^{[j]}} \subset Y$. Dado que Y y $X_{n^{[j]}}$ son generadas con la misma regla de división, $X_{n^{[j]}}$ pertenece a una subdivisión uniforme de Y. De acuerdo con la propiedad 3, $\underline{F}(Z, X_{n^{[j]}})$ es un límite inferior válido de $f^{[j]}$ sobre $(T^{[j]}, Y)$.

- $X_{n^{[j]}} \cap Y \neq \emptyset$, $Y \not\subseteq X_{n^{[j]}}$ and $X_{n^{[j]}} \not\subset Y$ no puede ocurrir porque todos los intervalos son generados usando la misma regla de división.

■

Esta propiedad establece que dado un intervalo Y de la variable común, la subcaja $\Psi^{[j]}(Y)$ es un límite inferior válido para la subfunción j en el rango de la variable común Y. Esta caja es de especial interés puesto que nos permitirá obtener un límite inferior de f a partir de las subcajas de cada subfunción $f^{[j]}$ que compartan valores en Y.

A continuación se describe como obtener un límite inferior de la función f para un subproblema $X \in S^{[j]}$. Consideraremos que f es evaluada sobre un problema (extendido) con los componentes de $I^{[j]}$ limitados al subproblema X, siendo libres en S los otros componentes $I \setminus I^{[j]}$.

Definición 21 *Dado un subproblema $X \in S^{[j]}$, definimos su extensión a S como $EX^{[j]} = \{E \in \mathbb{I}^n \mid E \subseteq S,\ E^{[j]} = X\}$ o de otra manera $EX^{[j]} = \{x \in S \mid x_m \in X_m,\ m \in I^{[j]};\ x_l \in S_l,\ l \notin I^{[j]}\}$.*

Proposición 5 *Consideremos la subfunción j y un subproblema $X \in \Lambda^{[j]} \cup \Omega^{[j]}$. Se puede obtener un límite inferior de f sobre $EX^{[j]}$ mediante el cálculo de*

$$\underline{F}^{[j]}(X) + \sum_{k=1,k\neq j}^{p} lb^{[k]}. \tag{6.6}$$

Demostración. Basándonos en (6.4), la propiedad 5 y la isotonía de la función de inclusión (propiedad 1),

$$lb^{[k]} = \min_{X \in \Lambda^{[k]} \cup \Omega^{[k]}} \underline{F}^{[k]}(X) \leq f^{[k]}(T^{[k]}, X_{n^{[j]}}). \tag{6.7}$$

Así,

$$\underline{F}^{[j]}(X) + \sum_{k=1,k\neq j}^{p} lb^{[k]} \leq f^{[j]}(X) + \sum_{k=1,k\neq j}^{p} f^{[k]}(T^{[k]}, X_{n^{[j]}}) = f(EX^{[j]}). \tag{6.8}$$

■

Se puede obtener un mejor límite inferior de f sobre $(EX^{[j]})$ mediante el uso del siguiente teorema.

Teorema 3 *Consideremos el subproblema $X \in \Lambda^{[j]} \cup \Omega^{[j]}$.*

$$\underline{F}_{DS}(EX^{[j]}) := \underline{F}^{[j]}(X) + \sum_{k=1,k\neq j}^{p} \underline{F}^{[k]}(\Psi^{[k]}(X_{n^{[j]}})) \tag{6.9}$$

es un límite inferior válido de f sobre $EX^{[j]}$.

Demostración. Este teorema es consecuencia directa de las proposiciones 4 y 5. ■

6.3.2 Rechazo

La siguiente propiedad establece que una caja puede rechazarse si no existe otra caja en el resto de subfunciones cuya variable común se solape.

Propiedad 7 *El subproblema* $X \in \Lambda^{[j]} \cup \Omega^{[j]}$ *puede eliminarse si* $\exists k \neq j,\ \forall Y \in \Lambda^{[k]} \cup \Omega^{[k]},\ X_{n^{[j]}} \cap Y_{n^{[k]}} = \emptyset$.

Se puede extender el test de rango a las subfunciones, para ello se define una cota superior del mínimo de cada subfunción $f^{[j]}$ en $X \subseteq S^{[j]}$ mediante la evaluación de las subfunciones en el punto medio de la variable común x_n, $x_n = m(X_{n^{[j]}})$. Sea $g^{[j]}(x_n)$ el mejor valor obtenido de $\overline{F}^{[j]}(m(X))$ para una caja $X \in \Lambda^{[j]} \cup \Omega^{[j]}$ con un valor de la variable común $m(X_{n^{[j]}}) = x_n$. El concepto de test de rango a las subfunciones se muestra en el siguiente teorema.

Teorema 4 *Sea el intervalo* $X \in S^{[j]}$. *Si* $\underline{F}^{[j]}(X) > g^{[j]}(x_n)$ *con* $x_n \in X_{n^{[j]}}$, *entonces* X *no puede contener una parte* $n^{[j]}$*-dimensional del óptimo global* x^*.

La figura 6.1 ilustra el teorema anterior para una subfunción j con dos variables, X_j y la variable común $X_{n^{[j]}}$. La estructura $g^{[j]}(x_n)$ almacena los mejores valores de los puntos medios para las subcajas evaluadas. Una caja X puede ser rechazada si su límite inferior es mayor que los valores almacenados en g cuyos puntos medios estén incluidos en la variable común de X.

Por tanto, este teorema nos permitirá eliminar de forma rigurosa subproblemas que no contienen una solución óptima, basándonos en una cota superior del mínimo global obtenida en cada una de las subfunciones.

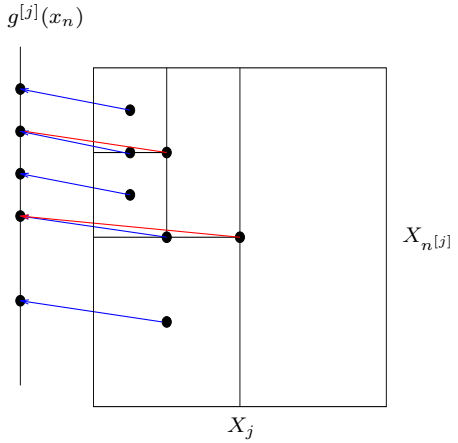

Figura 6.1: Test de rango para las subfunciones - La estructura $g^{[j]}(x_n)$ almacena los mejores valores de $\overline{F}^{[j]}(m(X))$. Una caja X puede ser rechazada si $\underline{F}^{[j]}(X) > g^{[j]}(x_n)$ con $x_n \in X_{n^{[j]}}$.

6.3.3 Monotonía

La propiedad de monotonía para las variables no comunes se extiende automáticamente de la propiedad 4 (página 20) y es aplicable a cada subfunción de forma separada.

Propiedad 8 *Monotonía de las variables no comunes. La propiedad 4 se aplica a las variables no comunes.*

Propiedad 9 *Monotonía de las variables comunes. Por la propiedad 4, centrándonos en la variable común x_n, sabemos que para un punto interior óptimo*

$$\frac{\partial f}{\partial x_n} = \sum_j \frac{\partial f^{[j]}}{\partial x_n} = 0. \tag{6.10}$$

De forma similar a la acotación (véase la sección 6.3.1), se debe mantener un seguimiento sobre lo que ocurre en las otras listas para comprobar la monotonía de la variable común. Introducimos la siguiente notación para la inclusión de la derivada:

$$\underline{\Theta}^{[j]}(Y) = \text{mín}\{\underline{F'}^{[j]}_{n^{[j]}}(X) \,|X \in \Lambda^{[j]} \cup \Omega^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\} \tag{6.11}$$

y

$$\overline{\Theta}^{[j]}(Y) = \text{máx}\{\overline{F'}^{[j]}_{n^{[j]}}(X) \,|X \in \Lambda^{[j]} \cup \Omega^{[j]},\ X_{n^{[j]}} \cap Y \neq \emptyset\}. \tag{6.12}$$

Mediante el siguiente teorema podemos formular la propiedad de monotonía para la variable común:

Teorema 5 *Consideremos el subproblema $X \subset S^{[j]}$. X no puede contener una parte $n^{[j]}$-dimensional de un punto interior óptimo x^* si*

$$\underline{F'}^{[j]}_{n^{[j]}}(X) + \sum_{k=1,k\neq j}^{p} \underline{\Theta}^{[k]}(X_{n^{[j]}}) > 0 \tag{6.13}$$

o

$$\overline{F'}^{[j]}_{n^{[j]}}(X) + \sum_{k=1,k\neq j}^{p} \overline{\Theta}^{[k]}(X_{n^{[j]}}) < 0. \tag{6.14}$$

Demostración. El teorema sigue de (6.10) teniendo en cuenta que, o bien $\sum_j \frac{\partial f^{[j]}}{\partial x_n} < 0$, o bien $\sum_j \frac{\partial f^{[j]}}{\partial x_n} > 0$. ■

6.4 Algoritmo basado en el enfoque DS (IBB-DS)

Las propiedades desarrolladas para la perspectiva DS en la sección 6.3 se pueden utilizar para implementar un nuevo algoritmo de ramificación y acotación que sirva para optimizar problemas aditivamente separables con variables comunes. A continuación se describen las diferentes reglas del algoritmo de ramificación y acotación para resolver el problema de optimización global cuando la función objetivo tiene las características de función aditivamente separable (véase la definición 19). Denotaremos este algoritmo como IBB-DS.

6.4.1 Regla de selección

Esta regla determina qué subproblemas serán visitados, tratando de buscar en las regiones más prometedoras. En primer lugar, hay que decidir en cual de las listas $\Lambda^{[j]}$ buscar. Se utilizará un esquema cíclico (*round robin*) entre las listas no vacías, aunque también pueden considerarse otros esquemas. Para seleccionar un subproblema X de la lista $\Lambda^{[j]}$ que tiene el turno de selección, se evaluaron los siguientes criterios:

1. El subproblema X con el menor valor de $\underline{F}^{[j]}(X)$.
2. El subproblema X con la mayor anchura $w(X)$.
3. El subproblema X con el menor valor almacenado de $\underline{F}_{DS}(EX^{[j]})$.
4. El subproblema X con la mayor anchura de $X_{n^{[j]}}$ y, como segundo criterio, aquel con el menor valor de $\underline{F}_{DS}(EX^{[j]})$.
5. Tomamos X según 3. Si $\exists k \neq j$ con $w(\Psi^{[k]}(X_{n^{[j]}})) > w(X)$, seleccionamos $\Psi^{[k]}(X_{n^{[j]}})$ en su lugar.
6. Tomamos X según 3. Evaluamos $\underline{F}_{DS}(EX^{[j]})$. Si su valor es mayor que el valor almacenado lo actualizamos y volvemos a almacenar X. Repetimos 6 hasta que no se pueda actualizar $\underline{F}_{DS}(EX^{[j]})$.

El criterio 1 usualmente no mejora la cota superior del mínimo global f^* debido a que se centra en los límites inferiores de las subfunciones en lugar de la función objetivo compuesta.

El criterio 2 es una búsqueda en anchura que intenta mejorar los valores de $\underline{F}_{DS}(EX^{[j]})$ (véase (6.9)) disminuyendo la anchura de los subproblemas y evitando grandes diferencias en el tamaño de los mismos en las diferentes listas.

El criterio 3 parece una buena opción ya que se basa en el criterio primero el mejor y usa información de la función compuesta, similar a la usada en el algoritmo estándar. Sin embargo, típicamente cada subfunción tiene un comportamiento diferente. Se diferencian en el número total de subcajas evaluadas y en el número total de nodos en cada nivel del árbol de búsqueda. Por tanto, es difícil seleccionar la caja que mejore $\underline{F}_{DS}(EX^{[j]})$ en el futuro, porque $\underline{F}_{DS}(EX^{[j]})$ no solo depende de X, sino también de $\Psi^{[k]}(X_{n^{[j]}})$ (véase (6.5) y (6.9)), que se actualiza durante la ejecución del algoritmo.

El criterio 4 combina los criterios 2 y 3 con sus ventajas y desventajas.

El criterio 5 intenta mejorar el valor de $\underline{F}_{DS}(EX^{[j]})$ dividiendo la caja más ancha de entre $X^{[j]}$ y $\Psi^{[k]}(X_n^{[j]})$.

El criterio 6 intenta mantener los valores de $\underline{F}_{DS}(EX^{[j]})$ actualizados. De acuerdo con nuestros experimentos preliminares, este es el mejor criterio de todos los presentados en esta sección, a pesar de que supone un coste adicional en tiempo. En la sección 6.4.6 explicaremos cómo hemos realizado la implementación del algoritmo para no comprometer demasiado el rendimiento en las búsquedas de $\Psi^{[k]}(X_n^{[j]})$.

6.4.2 Regla de división

Aplicaremos la bisección por el punto medio de la componente más ancha del subproblema $X \in S^{[j]}$. Se puede usar cualquier regla de división de las estudiadas en la sección 3.3.2 que genere subcajas de forma regular en la variable común.

6.4.3 Regla de acotación

Se obtiene un límite inferior de $f^{[j]}(X)$ por la ecuación (6.4) y un límite inferior de $f(X)$ por la ecuación (6.9). Se mantiene una lista de cotas superiores $g^{[j]}(x_n)$ para

aquellos valores de x_n tal que $\exists Y \in \Lambda^{[j]} \cup \Omega^{[j]}$ con $m(Y_{n^{[j]}}) = x_n$.

La cota superior del mínimo $\overline{f^*}$ habitualmente se actualiza evaluando f en un punto x. La cuestión es cómo seleccionar x para un intervalo $X = (Z, X_{n^{[j]}}) \subset S^{[j]} \neq S$. Los componentes de las variables no comunes pueden tomarse como el punto medio de Z y la parte no común de las otras subfunciones del correspondiente $\Psi^{[k]}(X_{n^{[j]}}),\ k \neq j$. Para el valor de la variable común x_n, debemos tomar cualquier punto que se solape (intersecte) entre X y $\Psi^{[k]}(X_{n^{[j]}})$, normalmente el punto medio de la intersección.

6.4.4 Regla de eliminación

Los siguientes criterios pueden aplicarse para rechazar los subproblemas $X = (Z, X_{n^{[j]}}) \in \Lambda^{[j]} \cup \Omega^{[j]}$ que no pueden contener componentes de la solución óptima.

Test del rango (RUpT): X es eliminada si $\underline{F_{DS}}(EX^{[j]}) > \overline{f^*}$.

Test del rango para subfunción (SubRUpT): Se elimina X si $\underline{F^{[j]}}(X) > g^{[j]}(x_n)$ con $x_n \in X_{n^{[j]}}$ (véase el teorema 4, página 86).

Valor no compartido de la variable común (UVCV): X es eliminada si $\exists k \neq j$ $\Psi^{[k]}(X_{n^{[j]}}) = \emptyset$ (véase la propiedad 7, página 86).

Test de monotonía de las variables no comunes (NCVMT): X es eliminada si $0 \notin F'^{[j]}_i(X),\ i \neq n^{[j]}$ y Z es interior con respecto a $T^{[j]}$ (véase la propiedad 8, página 87).

6.4.5 Regla de finalización

Se utiliza el criterio basado en la anchura de las subcajas (véase la sección 3.3.5). Todas las subcajas $X \in S^{[j]}$ con $w(X) \leq \epsilon$ se almacenan en $\Omega^{[j]}$. Al finalizar el algoritmo, se combinan todas las listas de subcajas finales $\Omega^{[j]},\ j = 1, \ldots, p$, en una lista de cajas finales Ω en el espacio n-dimensional. Dada la regla de división, el algoritmo tiene un número finito de pasos y dado que los subproblemas que no contienen componentes de la solución óptima son eliminados, el algoritmo converge a un conjunto finito de combinaciones que comparten el mismo rango de la variable común y que pueden contener las soluciones óptimas.

6.4.6 Otros aspectos de la implementación

Uno de los pasos que más tiempo consumen y que más influye en el rendimiento del algoritmo es la búsqueda de $\Psi^{[k]}(X_{n^{[j]}})$ para obtener $\underline{F}_{DS}(EX^{[j]})$ en la ecuación (6.9), y que además es utilizado por la regla de selección. Para evitar la búsqueda en las estructuras de datos $\Lambda^{[j]}$ y $\Omega^{[j]}$, que normalmente contienen muchos elementos, se utiliza una estructura de datos auxiliar $M^{[j]}$. $M^{[j]}$ es una lista de $\Psi^{[j]}(X_{n^{[j]}})$ para todos los intervalos $X_{n^{[j]}}$ existentes.

Además, el algoritmo IBB-DS tiene tres fases bien diferenciadas:

- Fase 1: Se ejecuta el algoritmo de ramificación y acotación basado en las reglas descritas. Al finalizar se obtendrán las listas de subcajas finales $\Omega^{[j]}$.
- Fase 2: El valor de $\underline{F}_{DS}(EX^{[j]})$ es actualizado para todas las subcajas $X \in \Omega^{[j]}$. El valor actual de $\underline{F}_{DS}(EX^{[j]})$ puede estar basado en una subcaja almacenada $\Psi^{[k]}(X_{n^{[j]}})$ anterior a una nueva caja con un valor mejor. El objetivo es actualizar todos los límites inferiores para eliminar subcajas mediante los test RUpT y UVCV.
- Fase 3: Se combinan las subcajas en $\Omega^{[j]},\ j = 1, \ldots, p$ que tienen el mismo rango de la variable común para obtener la lista Ω de cajas en la dimensión completa. Se aplican los test RUpT y de monotonía a estas cajas n-dimensionales.

6.5 Evaluación experimental del algoritmo de optimización global para funciones aditivamente separables con una variable común

En primer lugar discutiremos las características de los experimentos diseñados y después mostraremos y discutiremos los resultados numéricos obtenidos.

6.5.1 Diseño de los experimentos

Se ha creado un conjunto de 13 funciones de prueba con una variable común. El apéndice D describe las características de estas funciones. Las cinco primeras son

tridimensionales, lo que facilita la obtención de resultados gráficos similares a la figura 6.2. La función Ejemplo 1 es una función simple de prueba creada específicamente para esta investigación. La mayoría de los problemas se han construido combinando dos funciones de optimización global bien conocidas, compartiendo una de sus variables. Por ejemplo, el problema N. 3 (L5P) es la combinación de las funciones Levy 5 y Price, y el problema N. 6 (G5G5) es la combinación de dos funciones Griewank 5. Se han empleado dos valores diferentes para el criterio de terminación, $\epsilon = 10^{-3}$ y $\epsilon = 10^{-6}$. Cada una de las funciones de prueba se ha resuelto primero empleando el algoritmo 2 (IBB) y después el algoritmo presentado en la sección 6.4 (IBB-DS).

Se utiliza la función de inclusión isótona $F_{\cap}$ para calcular las cotas, véase la ecuación (6.15).

$$F_{\cap}(X) := F_{NIE}(X) \cap T_1(b, X), \tag{6.15}$$

donde $F_{NIE}(X)$ es la extensión natural a intervalos de f (véase la propiedad 1, página 19) y $T_1(b, X)$ es la forma de Baumann (véase la sección 2.5.2, página 20).

Para la realización de estos experimentos se ha utilizado un procesador Intel Core Duo T2300 1.66GHz con 1.5Gb de RAM.

6.5.2 Resultados numéricos

Primero analizaremos los resultados de una de las funciones de prueba para mostrar en detalle las ventajas y desventajas de ejecutar el algoritmo dimensional completo IBB comparado con el algoritmo IBB-DS.

La tabla 6.1 muestra el comportamiento de los algoritmos IBB e IBB-DS para la función de prueba N. 2 (GP3). En la parte superior se muestran los resultados de IBB y en la parte inferior los resultados de IBB-DS divididos en las fases 1, 2 y 3. Se han medido los siguientes indicadores de rendimiento:

- $|\Omega|$: el número de cajas finales n-dimensionales.
- $[lb, \overline{f^*}]$: intervalo final que incluye f^*.
- CPU: tiempo de ejecución en segundos.
- El valor de esfuerzo se ha medido de la siguiente manera:

- Para IBB = $\#F_{\cap} + n \cdot \#F'_{NIE}$, donde $\#$ es el número de evaluaciones.
- Para IBB-DS= $\#F_{\cap} + n \cdot \#F'_{NIE} + \sum_{i=1}^{p} \#F_{\cap}^{[i]} + n^{[j]} \cdot \#F^{[i]\prime}_{NIE}$.
  Nótese que la medida del esfuerzo del algoritmo IBB-DS es muy pesimista ya que el coste de evaluar $F_{\cap}$ es mayor que el de evaluar $F_{\cap}^{[i]}$.

Otra información mostrada en la tabla 6.1 es el número de cajas rechazadas por los test presentados en el algoritmo 2 y en la sección 6.4.4. Nótese que la fase 3 de IBB-DS usa los test de rechazo del algoritmo IBB. El número de cajas finales $|\Omega^{[j]}|$ y el intervalo que incluye f^* también se muestran para las fases 1 y 2. Cualquier $lb^{[j]}$ (véase (6.4)) puede usarse para obtener el intervalo que incluye f^* en las fases 1 y 2.

El rendimiento del algoritmo IBB-DS, en términos de esfuerzo y tiempo de CPU, es mejor que el algoritmo IBB para la función de prueba GP3. Además, IBB-DS obtiene una mejor inclusión del óptimo global f^* para $\epsilon = 10^{-3}$.

En primer lugar nos centraremos en las posibles ventajas de la perspectiva descompuesta:

1 La descomposición reduce la dimensión de los problemas.

2 La separación de los términos puede conducir a un menor número de ocurrencias de la misma variable. Esto ayuda a disminuir la sobrestimación de la aritmética de intervalos en el cálculo de la inclusión del rango real de la función.

3 Los nuevos test de rechazo para las subfunciones (SubRUpT, NCVMT y UVCV) funcionan bien debido al punto 2 y permiten rechazar subcajas que no pueden ser rechazadas por el algoritmo IBB.

4 En IBB-DS, el punto evaluado para intentar actualizar $\overline{f^*}$ esta restringido en la componente común a $X^{[j]} \cap \Psi^{[k]}(X_{n^{[j]}}),\ j \neq k$.

5 RUpT en la fase 1 del algoritmo IBB-DS no es tan efectivo como en el algoritmo IBB, pero aún rechaza un gran número de cajas.

Posibles desventajas que podemos derivar de las observaciones:

Tabla 6.1: Resultados obtenidos por los algoritmos IBB e IBB-DS para la función de prueba N. 2 (GP3)

Algoritmo IBB	$\epsilon = 10^{-3}$	$\epsilon = 10^{-6}$
RUpT	40 777	44 615
Monotonía	5 398	6 997
$\lvert\Omega\rvert$	141	140
$[lb, \overline{f^*}]$	$[64.994799, 65.000023]$	$[64.999999, 65.000001]$
CPU	6.80	7.61
Esfuerzo	416 836	465 760
Algoritmo IBB-DS fase 1		
RUpT	2 628	3 683
SubRUpT	886	1 334
UVCV	232	1 013
NCVMT	962	2 703
$\lvert\Omega^{[1]}\rvert$	162	147
$\lvert\Omega^{[2]}\rvert$	121	129
$[lb^{[1]}, \overline{f^*}]$	$[57.864553, 65.000023]$	$[62.408512, 65.000001]$
Algoritmo IBB-DS fase 2		
RUpT	18	20
UVCV	102	100
$\lvert\Omega^{[1]}\rvert$	79	75
$\lvert\Omega^{[2]}\rvert$	84	81
$[lb^{[1]}, \overline{f^*}]$	$[64.996493, 65.000023]$	$[64.999999, 65.000001]$
Algoritmo IBB-DS fase 3		
RUpT	42	35
Monotonía	0	0
$\lvert\Omega\rvert$	137	133
$[lb, \overline{f^*}]$	$[64.996493, 65.000002]$	$[64.999999, 65.000001]$
CPU	2.08	3.67
Esfuerzo	35 823	63 894

1 En la fase 1, el intervalo que incluye f^* es mayor que el intervalo final en la fase 3. El límite inferior de $f(EX^{[j]})$ obtenido por $\underline{F}_{DS}(EX^{[j]})$ en (6.9) no depende solamente de $F^{[j]}(X^{[j]})$ sino además de cuántas subcajas en $\Lambda^{[k]} \cup \Omega^{[k]},\ k \neq j$ tienen puntos en común con $X^{[j]}_{n^{[j]}}$. Estas subcajas determinan $\Psi^{[k]}(X^{[j]}_{n^{[j]}})$, véase (6.5). Pocas subcajas en $\Lambda^{[k]} \cup \Omega^{[k]}$ da mejores resultados de $\underline{F}_{DS}(EX^{[j]})$. Por tanto, en la fase 3 el RUpT es efectivo rechazando más cajas (tabla 6.1). Además, las conclusiones obtenidas en el capitulo anterior mostraron que la evaluación de la forma de Baumann separable podía ofrecer límites inferiores peores que la versión compuesta. Por ello, una versión multidimensional de la función $ASLB\varphi$ podría mejorar sustancialmente los resultados del algoritmo IBB-DS.

2 El test de monotonía en la variable común se deja para la fase 3, lo que hace que no se rechacen estas cajas hasta que el algoritmo prácticamente ha finalizado.

3 Mejores valores de $\underline{F}_{DS}(EX^{[j]})$ se obtienen cuando $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in \Omega^{[k]}$ (lista de cajas finales), si se comparan con los valores de $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in \Lambda^{[k]}$. La fase 2 mejora algunos valores $\underline{F}_{DS}(EX^{[j]})$, porque $\Psi^{[k]}(X^{[j]}_{n^{[j]}}) \in \Omega^{[k]}$. RUpT es más efectivo al final de la fase 2.

4 Trabajando con las cajas finales en la fase 2, UVCV es efectivo por las siguientes razones:

- En la fase 1, una subfunción j podría generar subcajas finales en $\Omega^{[j]}$, mientras que subcajas de otra subfunción k que comparte variable común son almacenadas en $\Lambda^{[k]}$. No todas estas subcajas en $\Lambda^{[k]}$ tienen descendientes que alcancen $\Omega^{[k]}$ por lo que estas subcajas almacenadas como finales en $\Omega^{[j]}$ no tienen su correspondiente $\Psi^{[k]}(X^{[j]}_{n^{[j]}})$, y por lo tanto podrían haber sido rechazadas en la fase 2 por el test UVCV.
- La eliminación de subcajas de una subfunción por RUpT en la fase 2 (véase el punto 2) permite al test UVCV eliminar subcajas de otras subfunciones.

La figura 6.2 muestra las subcajas rechazadas y finales de las fases 1 y 2 por el algoritmo IBB-DS en la función GP3 con una precisión de $\epsilon = 10^{-1}$. En esta figura se ilustran las ventajas y desventajas mencionadas anteriormente.

En las tablas 6.2 y 6.3, se mide la efectividad de los algoritmos IBB e IBB-DS en términos de inclusión del intervalo que contiene f^* y el número de cajas finales en $|\Omega|$ para los dos valores de precisión. Para ambos valores de ϵ el algoritmo IBB-DS obtiene un menor o igual número de cajas finales. Más aún, el algoritmo IBB-DS obtiene una inclusión mejor o igual de f^*.

Las tablas 6.4 y 6.5, miden la eficiencia en términos de esfuerzo, tiempo de CPU (en segundos) y máximo número de elementos en las listas alcanzado durante la ejecución. Para la precisión $\epsilon = 10^{-3}$, el algoritmo IBB-DS necesita menos esfuerzo que el algoritmo IBB para todas las funciones de prueba y emplea un tiempo similar o menor, excepto para el caso N. 9 (RB5G2). Para la precisión $\epsilon = 10^{-6}$, IBB-DS necesita menos esfuerzo que IBB para 6 de las 13 funciones de prueba y emplea menos tiempo en 4 de ellas. Para la mayoría de los problemas, una mayor precisión de ϵ conduce a un incremento en el tamaño de $\Lambda^{[j]}$ y $\Omega^{[j]}$. Esto hace que los test de rechazo del algoritmo IBB-DS no funcionen adecuadamente.

El número máximo de elementos en la lista de trabajo y en la lista final afecta directamente la eficiencia del algoritmo IBB-DS respecto al tiempo de CPU. Esto puede observarse directamente cuando utilizamos mayores precisiones. Como se mencionó en la desventaja número 1 en la página 95, un mayor número de elementos en las listas conduce a una peor estimación del límite inferior. Adicionalmente, se necesita bastante más esfuerzo computacional para encontrar el valor $\Psi^{[j]}(Y)$ definido en (6.5). Para realizar esto se tendrían que visitar completamente las listas de la otra subfunción, aunque en nuestra implementación hemos utilizado una estructura auxiliar $M^{[j]}$, tal como se explicó en la sección 6.4.6. Para la precisión $\epsilon = 10^{-3}$, el número máximo de elementos en la lista de trabajo y final es, en general, menor cuando usamos IBB-DS, si se compara con IBB.

Podemos observar que para el caso de mayores dimensiones, por ejemplo la función N. 6 (G5G5) con 9 variables, la ventaja de usar la perspectiva de descomposición en subfunciones es mayor que para los casos de menores dimensiones. Para una función tridimensional con una variable común, usar la perspectiva descompuesta significaría tener dos subfunciones de dos dimensiones, mientras que

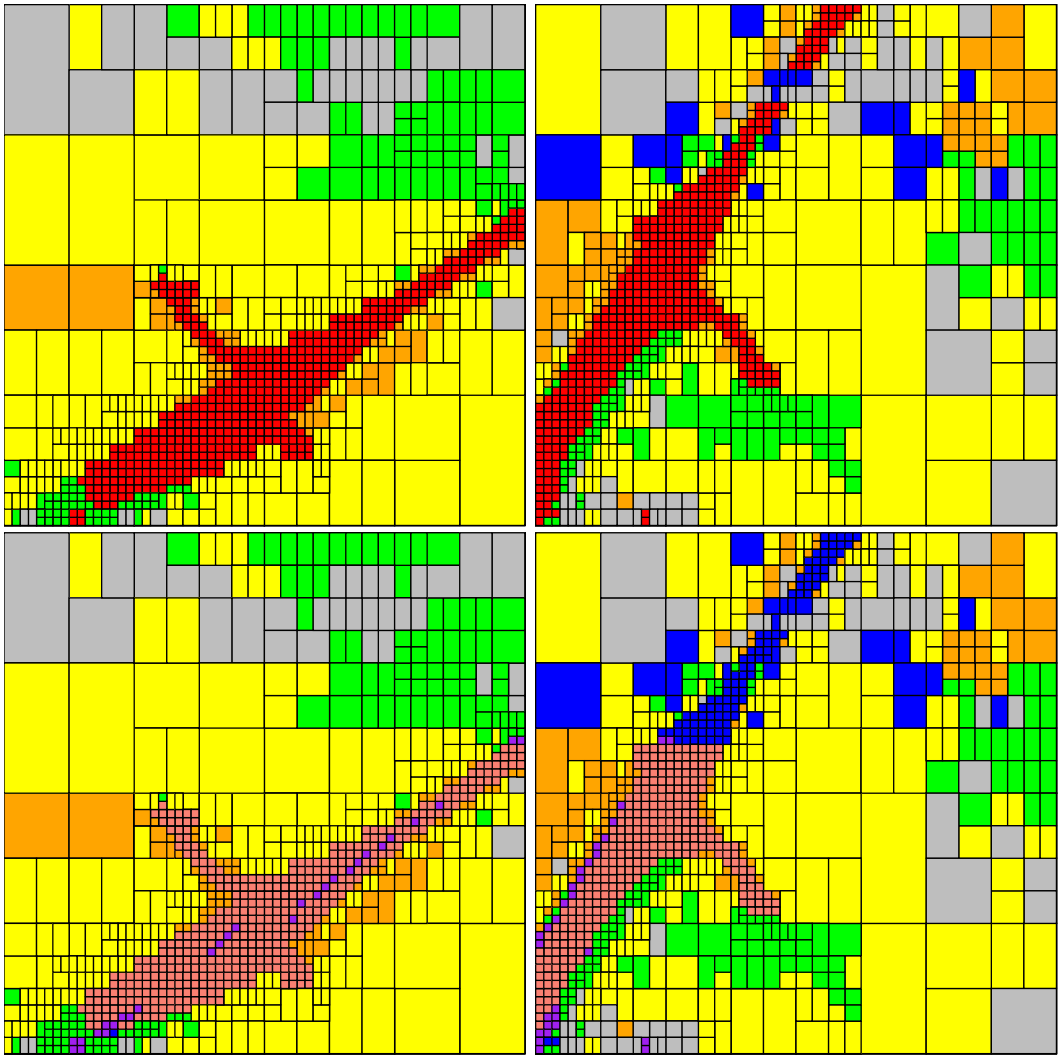

Figura 6.2: Ilustración de la ejecución del algoritmo IBB-DS para el problema GP3 - Las imágenes de la fila superior muestran las cajas en la fase 1 y las imágenes de la fila inferior muestran las cajas en la fase 2. La función $f^{[1]}(x_1, x_3)$ se muestra en el lado izquierdo y $f^{[2]}(x_2, x_3)$ en el derecho. El color de las subcajas muestra el éxito de los test de rechazo: RUpT fase 1 (test de rango (gris) y test de corte (verde)), SubRUpT (amarillo), NCVMT (naranja), UVCV fase 1 y 2 (azul), cajas finales en la fase 1 (rojo), RUpT fase 2 (violeta), cajas finales en la fase 2 (salmon). La precisión utilizada fue $\epsilon = 10^{-1}$.

Tabla 6.2: Efectividad de los algoritmos IBB e IBB-DS para $\epsilon = 10^{-3}$

			IBB		IBB-DS	
N.	n_1	n_2	$[lb, \overline{f^*}]$	$\lvert\Omega\rvert$	$[lb, \overline{f^*}]$	$\lvert\Omega\rvert$
1	2	2	$[-85.000001, -84.990234]$	4	$[-85.000001, -84.995117]$	4
2	2	2	$[64.994799, 65.000023]$	141	$[64.996493, 65.000002]$	137
3	2	2	$[-172.2774, -172.2768]$	3	$[-172.2774, -172.2768]$	3
4	2	2	$[-123.743285, -123.743157]$	2	$[-123.743285, -123.743159]$	2
5	2	2	$[-168.6570, -168.6566]$	24	$[-168.6570, -168.6566]$	24
6	5	5	$[-0.00001, 1.199041 \cdot 10^{-14}]$	512	$[-0.00001, 1.332268 \cdot 10^{-14}]$	512
7	6	3	$[-9.652299, -9.629085]$	4	$[-9.652299, -9.629121]$	4
8	5	3	$[-35.954858, -35.954755]$	9	$[-35.954858, -35.954780]$	9
9	5	2	$[0.153744, 0.154892]$	1366	$[0.153744, 0.154892]$	1258
10	4	4	$[-20.939446, -20.939333]$	1	$[-20.939446, -20.939340]$	1
11a	3	2	$[-0.000171, 7.578985 \cdot 10^{-6}]$	1101	$[-0.000129, 8.643538 \cdot 10^{-7}]$	569
11b	2	3	$[-0.000171, 7.578985 \cdot 10^{-6}]$	1101	$[-6.879867 \cdot 10^{-5}, 8.643538 \cdot 10^{-7}]$	67
12	3	3	$[-118.021834, -118.020600]$	2	$[-118.021834, -118.020827]$	2
13	3	3	$[-29.000001, -28.996337]$	16	$[-29.000001, -28.998168]$	16

Tabla 6.3: Efectividad de los algoritmos IBB e IBB-DS para $\epsilon = 10^{-6}$

			IBB		IBB-DS	
N	n_1	n_2	$[lb, \overline{f^*}]$	$\lvert\Omega\rvert$	$[lb, \overline{f^*}]$	$\lvert\Omega\rvert$
1	2	2	$[-85.000001, -84.999990]$	4	$[-85.000001, -84.999995]$	4
2	2	2	$[64.999999, 65.000001]$	137	$[64.999999, 65.000001]$	133
3	2	2	$[-172.2770, -172.2769]$	3	$[-172.2770, -172.2769]$	3
4	2	2	$[-123.743164, -123.743163]$	2	$[-123.743164, -123.743163]$	2
5	2	2	$[-168.6567, -168.6566]$	24	$[-168.6567, -168.6566]$	24
6	5	5	$[-0.0000001, 1.199041 \cdot 10^{-14}]$	512	$[-0.0000001, 1.332268 \cdot 10^{-14}]$	512
7	6	3	$[-9.652141, -9.652111]$	16	$[-9.652141, -9.652118]$	16
8	3	3	$[-35.954789, -35.954788]$	9	$[-35.954789, -35.954788]$	9
9	5	2	$[0.154870, 0.154871]$	1628	$[0.154870, 0.154871]$	1628
10	4	4	$[-20.939350, -20.939349]$	2	$[-20.939350, -20.939349]$	2
11a	3	2	$[-1.630625 \cdot 10^{-10}, 7.227997 \cdot 10^{-12}]$	1107	$[-1.223355 \cdot 10^{-10}, 8.242295 \cdot 10^{-13}]$	581
11b	2	3	$[-1.630625 \cdot 10^{-10}, 7.227997 \cdot 10^{-12}]$	1107	$[-6.561152 \cdot 10^{-11}, 8.242295 \cdot 10^{-13}]$	67
12	3	3	$[-118.020864, -118.020863]$	2	$[-118.020864, -118.020863]$	2
13	3	3	$[-29.000001, -28.999996]$	1	$[-29.000001, -28.999998]$	1

para una funcion de 9 dimensiones con una variable común tendríamos dos subfunciones de cinco dimensiones cada una. Por otro lado, la mayor desventaja de usar descomposición la encontramos en la función N. 7 (HPHM4) para $\epsilon = 10^{-6}$. Este caso se caracteriza por un desbalanceo en el número de variables de las subfunciones. Esto causa un desbalanceo en el tamaño de las listas de las subfunciones. Para este tipo de casos debería estudiarse una mejor regla de selección que disminuya la diferencia en el progreso de los algoritmos de ramificación y acotación para cada subfunción.

Por último queremos destacar en las tablas 6.2-6.5 que dos descomposiciones diferentes del mismo problema (función N. 11 (Colville)) produce resultados numéricos diferentes (función N. 11b obtiene mejor eficiencia que N. 11a). Esto muestra que la forma en la que una función es descompuesta afecta la eficiencia, dependiendo de los tamaños máximos de las listas que se obtengan.

6.6 Conclusiones

En este capítulo se han estudiado las funciones separables con variables comunes en las subfunciones generadas. Para responder a la pregunta de si es posible utilizar la estructura separable de las funciones para obtener algoritmos de ramificación y acotación más eficientes, se ha desarrollado toda una serie de propiedades desde una persepectiva de descomposición en subfunciones, que permita obtener cotas inferiores de la función objetivo (compuesta) y rechazar cajas en las subfunciones que no contienen el óptimo global. Por tanto, se han sentado las bases para el desarrollo de algoritmos de optimización específicos para funciones aditivamente separables. También se ha mostrado que se puede extender fácilmente el test de monotonía de las variables individuales (no comunes). Para aplicar el test de monotonía de la variable común sobre todas las subfunciones se necesita una investigación más profunda de las posibles implementaciones.

Se han incluido las propiedades desarrolladas para las funciones aditivamente separables con variables comunes en un nuevo algoritmo de ramificación y acotación, IBB-DS, como parte de las reglas de selección, acotación y eliminación.

Los resultados experimentales obtenidos al resolver las funciones separables desde la perspectiva DS muestran una mejora potencial en el esfuerzo y tiempo

Tabla 6.4: Eficiencia de los algoritmos IBB e IBB-DS para $\epsilon = 10^{-3}$. El carácter $*$ representa la ganancia del algoritmo IBB-DS.

	IBB				IBB-DS						ganancia	
N	Esfuerzo	CPU	$\lvert\Lambda\rvert^U$	$\lvert\Omega\rvert^U$	Esfuerzo	CPU	$\lvert\Lambda^{[1]}\rvert^U$	$\lvert\Omega^{[1]}\rvert^U$	$\lvert\Lambda^{[2]}\rvert^U$	$\lvert\Omega^{[2]}\rvert^U$	Esfuerzo	CPU
1	1 972	0.02	35	4	1 273	0.02	14	4	27	10	$*$	$*$
2	416 836	6.80	6 411	141	35 823	2.08	709	162	479	121	$*$	$*$
3	5 293	0.13	65	3	3 346	0.08	68	12	64	13	$*$	$*$
4	17 389	0.42	368	2	5 078	0.14	86	39	88	32	$*$	$*$
5	51 814	1.25	1 038	24	9 737	0.26	163	72	110	102	$*$	$*$
6	1 946 092	33.88	512	512	89 643	1.50	32	32	32	32	$*$	$*$
7	14 954	0.32	50	4	12 913	0.28	28	58	86	48	$*$	$*$
8	16 386 891	514.44	251 654	9	608 927	426.95	178	10 936	11 248	36	$*$	$*$
9	270 676	2.85	1 246	1 366	148 902	4.98	887	979	101	1 044	$*$	
10	3 299	0.14	83	1	1 946	0.06	19	3	18	5	$*$	$*$
11a	357 149	2.46	2 244	1 101	65 286	2.43	382	409	119	879	$*$	$*$
11b	357 149	2.46	2 244	1 101	6 920	0.09	43	62	29	9	$*$	$*$
12	54 952	0.50	568	2	7 020	0.11	94	43	92	16	$*$	$*$
13	9 946	0.07	33	16	2 572	0.03	17	5	12	5	$*$	$*$

Tabla 6.5: Eficiencia de los algoritmos IBB e IBB-DS para $\epsilon = 10^{-6}$. El carácter $*$ representa la ganancia del algoritmo IBB-DS.

	IBB				IBB-DS						ganancia	
N	Esfuerzo	CPU	$\lvert\Lambda\rvert^U$	$\lvert\Omega\rvert^U$	Esfuerzo	CPU	$\lvert\Lambda^{[1]}\rvert^U$	$\lvert\Omega^{[1]}\rvert^U$	$\lvert\Lambda^{[2]}\rvert^U$	$\lvert\Omega^{[2]}\rvert^U$	Esfuerzo	CPU
1	3 187	0.02	55	4	2 064	0.03	14	4	27	13	$*$	
2	465 760	7.61	6 411	140	63 894	3.67	709	147	479	129	$*$	$*$
3	6 274	0.16	65	3	27 137	1.02	96	328	356	378		
4	17 866	0.44	368	2	52 360	3.83	392	795	620	660		
5	58 132	1.40	1 038	24	171 670	14.42	2 232	2 412	778	1 931		
6	2 913 772	50.68	512	512	131 243	2.15	32	32	32	32	$*$	$*$
7	30 439	0.64	59	16	594 826	92.76	968	2 147	2 762	6 644		
8	16 392 624	515.60	251 654	9	682 367	444.22	570	11 070	11 246	810	$*$	$*$
9	1 623 166	17.0	3 720	1 628	5 665 430	7 329.00	26 475	28 032	2 007	23 882		
10	4 523	0.19	83	2	6 599	0.21	19	23	18	26		
11a	821 954	5.68	2 298	1 107	148 783	12.41	382	414	400	2 154	$*$	
11b	821 954	5.68	2 298	1 107	10 455	0.13	42	62	48	9	$*$	$*$
12	56 252	0.51	568	2	72 593	3.40	446	553	443	521		
13	16 901	0.12	41	1	4 354	0.05	17	5	12	5	$*$	$*$

de CPU, respecto del algoritmo estándar IBB. Esta mejora obtenida por el algoritmo IBB-DS depende principalmente del número de subcajas almacenadas durante la ejecución del algoritmo. Los resultados demuestran las posibles ventajas de usar una estrategia de descomposición frente a un estrategia dimensional completa cuando la precisión requerida de los resultados no es alta.

El cálculo de límites inferiores separables que mejoren sustancialmente los valores obtenidos por el algoritmo IBB-DS mejoraría sustancialmente su eficiencia. En esta línea de trabajo, la investigación presentada en el capítulo 5 sobre acotación de funciones aditivamente separables resulta muy prometedora. Además, una cuestión interesante sería aplicar la teoría desarrollada en este capítulo, en problemas reales donde la función objetivo tenga las características de separabilidad que se han estudiado.

Por último, se han sentado las bases para el desarrollo de nuevos métodos y estrategias que permitan mejorar la eficiencia de los algoritmos de optimización para funciones aditivamente separables con variables comunes.

Me interesa el futuro porque es el sitio donde voy a pasar el resto de mi vida.

Woody Allen

CAPÍTULO

7

Conclusiones y trabajo futuro

Los métodos de ramificación y acotación intervalares son herramientas que permiten resolver los problemas de optimización global de forma rigurosa. Los problemas de optimización aparecen con frecuencia en ámbitos como la Ingeniería, la Economía, la Matemática, la Química, etc. El cálculo de la mejor solución posible para un determinado problema puede suponer un ahorro en costes de producción o un incremento de las ganancias o los beneficios. A diferencia de otros métodos, los algoritmos de ramificación y acotación intervalares siempre obtienen el óptimo global.

Se han presentado los estudios más relevantes relacionados con la temática de esta tesis. Todos estos trabajos van orientados a mejorar el rendimiento de estos métodos debido al enorme coste computacional que es necesario para resolver un problema de optimización global, que pertenece a la categoría NP-Completo. El objetivo de esta tesis ha ido orientado a mejorar el rendimiento de estos métodos en tres ámbitos: la predicción de la carga de trabajo, el cálculo de cotas más precisas del rango real de la función objetivo en un intervalo y métodos específicos más eficientes para funciones aditivamente separables con una variable común.

La predicción de la carga computacional, tal y como se ha realizado en esta tesis, no había sido abordada hasta el momento. Los trabajos previos, en su mayoría, consideran a los nodos con mejores límites inferiores más prometedores, y por tanto el reparto de nodos entre procesadores se basa en esa heurística. Un factor importante cuando se trabaja con algoritmos paralelos es determinar el tiempo de cómputo restante para poder determinar de forma eficiente los recursos computacionales en cada momento de la ejecución. Es habitual que los algoritmos paralelos empeoren su *speed-up* en cierto momento, a medida que se incrementa el número de procesadores para tratar de disminuir aún más el tiempo de ejecución. Esto se debe a que la carga de trabajo no es suficiente para que todos los procesadores se encuentren ocupados. Por tanto, se hace necesario que la asignación de recursos no sea una decisión que se tome a priori, antes de comenzar la ejecución del algoritmo, sino que sea el propio algoritmo el que decida los recursos computacionales que necesita en cada momento. La investigación presentada en el capítulo 4 propone tres métodos para estimar la cantidad de nodos que restan por evaluar en el algoritmo de ramificación y acotación, lo que es una medida real del tiempo que resta para finalizar la ejecución. Todos los métodos propuestos se basan en que el factor de rechazo que ha presentado el algoritmo de ramificación durante su ejecución, se mantendrá en el futuro. Esto no tiene porqué ser cierto en todos los casos, aunque los resultados muestran una buena estimación en las etapas medias y últimas para el conjunto de funciones de prueba. Entre los distintos métodos, ILLEM es el único que provee información individual para cada nodo, lo que es de gran utilidad a la hora de utilizar estrategias de balanceo dinámico de la carga en algoritmos paralelos de optimización global.

El cálculo de cotas o límites del rango real de la función objetivo que provee la aritmética de intervalos mediante las funciones de inclusión es una pieza clave para un buen rendimiento del algoritmo de ramificación y acotación intervalar. Estas cotas serán mejores cuanto más se asemejen a las cotas reales. El capítulo 5 presenta una nueva función de inclusión basada en términos aditivamente separables, $ASLB_{\varphi}$, y se demuestra matemáticamente que las cotas obtenidas son mejores que las formas LBVF. Estas mejoras no permiten reducir el coste computacional de los algoritmos de optimización global intervalar porque no son suficientes para permitir rechazos adicionales de nodos. Se espera que en estudios futuros donde

se aborden problemas de más de una dimensión, que pueden verse afectados por una mayor sobrestimación en la función de inclusión, el método n-dimensional de $ASLB_\varphi$ permita reducir la complejidad computacional de este tipo de algoritmos.

Los métodos de ramificación y acotación intervalares deben explotar las características y cualidades propias de cada problema de optimización si quieren obtener el máximo rendimiento posible. En el capítulo 6 se estudió uno de esos problemas específicos, las funciones aditivamente separables con una variable común, y se presentó un nuevo método para resolverlos de manera más eficiente. Los resultados experimentales muestran que es posible disminuir el esfuerzo y el número de iteraciones necesarias para hallar la solución de este tipo de problemas cuando la precisión empleada no es elevada. En cambio, estas mejoras no aparecen cuando aumenta la precisión requerida en las soluciones. Esto es debido a que el límite inferior global de un nodo es mejor cuanto menor es el número de nodos que hay en la lista de trabajo y final de la otra subfunción que comparten total o parcialmente la variable común. Se ha demostrado experimentalmente que este número de nodos crece con el aumento de la precisión requerida. En este ámbito aún quedan por abordar como trabajo futuro los siguientes estudios: La mejora de las cotas obtenidas en las subfunciones mediante el uso de la función $ASLB_\varphi$ presentada en el capítulo 5, y el uso de una estrategia de selección que permita obtener un número similar de nodos en las listas de trabajo para cada subfunción. Además, el algoritmo propuesto tiene unas características adecuadas para ser paralelizado.

En definitiva, la tesis presentada aquí realiza el estudio en tres ámbitos de los algoritmos de optimización global intervalar: predicción, acotación y separabilidad, estableciendo las bases para el desarrollo de nuevos métodos que mejoren la eficiencia de este tipo de algoritmos, tanto secuenciales como paralelos.

APÉNDICE

A

Publicaciones derivadas de esta tésis

Este apéndice contiene la lista de publicaciones (extraída de la bibliografía) derivadas de la investigación desarrollada durante la elaboración de esta tésis. Se han añadido además los indicios de calidad de cada una de estas publicaciones y dentro de cada sección aparecen ordenadas por año de publicación (primero las más antiguas).

A.1 Artículos en revistas internacionales

[9] Berenguel, J.L., Casado, L.G., García, I. & Hendrix, E.M.T. (2011). On estimating workload in interval branch-and-bound global optimization algorithms. *Journal of Global Optimization*, 1–24, DOI:10.1007/s10898-011-9771-5.

⋆ El índice de impacto para esta revista según el JCR 2011 es de 1.186. En la categoría *Operations Research & Management* ocupa la posición 24/77 y en la categoría *Mathematics Applied* en la 53/245.

[11] Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T. & Messine, F. (2012). On interval branch-and-bound for additively separable functions with common variables. *Journal of Global Optimization*, 1–21, DOI:10.1007/s10898-012-9928-x.

⋆ El índice de impacto para esta revista según el JCR 2011 es de 1.186. En la categoría *Operations Research & Management* ocupa la posición 24/77 y en la categoría *Mathematics Applied* en la 53/245.

A.2 Artículos en congresos internacionales en WOK o SCOPUS

[10] Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T. & Messine, F. (2012). On lower bounds using additively separable terms in interval B&B. In B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. Rocha, D. Taniar & B. Apduhan, eds., *Computational Science and Its Applications – ICCSA 2012*, vol. 7335 of *Lecture Notes in Computer Science*, 119–132, Springer Berlin / Heidelberg, DOI:10.1007/978-3-642-31137-6_9.

A.3 Otros artículos en congresos internacionales

[8] Berenguel, J.L., Casado, L.G., Hendrix, E.M.T., Messine, F. & García, I. (2010). On interval branch-and-bound algorithm for additively separable functions with one common variable. In *TOGO10: Proceedings of the Toulouse Global Optimization Workshop*, 23–26.

[12] Berenguel, J.L., Casado, L.G., Hendrix, E.M.T., Messine, F. & García, I. (2012). On lower bounds using separable terms in interval B&B for one-dimensional problems. In *NAGO12: Proceedings of Global Optimization Workshop 2012*, 39–42.

APÉNDICE

B

Nodos por nivel y secuencia-γ de las funciones de prueba

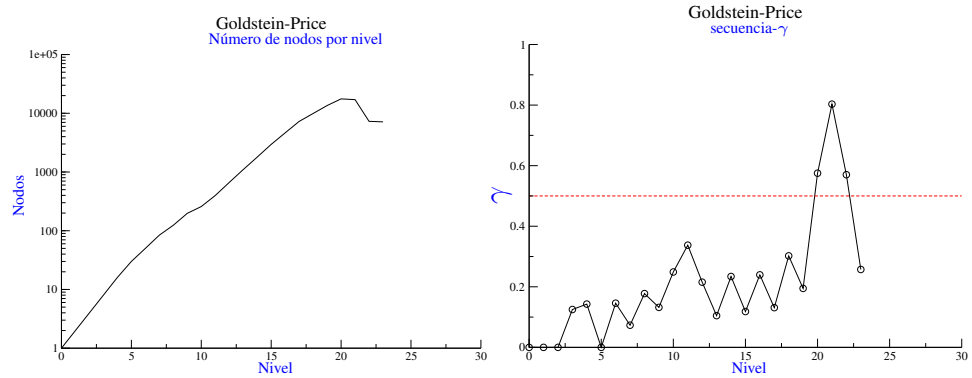

Figura B.1: Número de nodos por nivel y secuencia-γ para el problema Goldstein-Price - Precisión $\epsilon = 10^{-3}$.

B. NODOS POR NIVEL Y SECUENCIA-γ DE LAS FUNCIONES DE PRUEBA

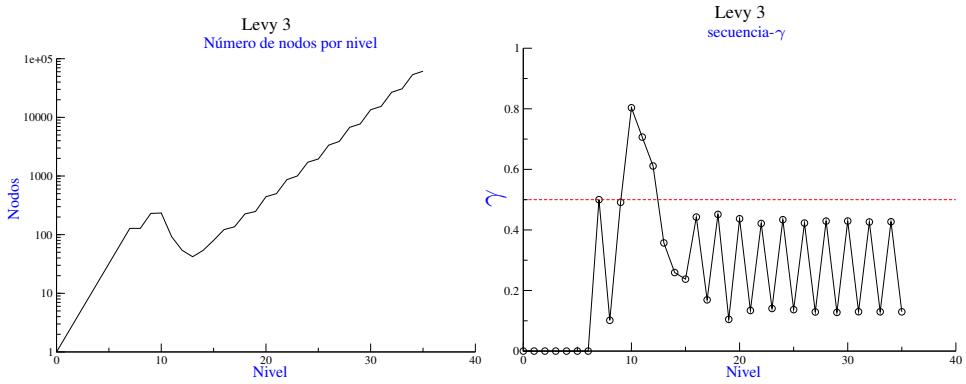

Figura B.2: Número de nodos por nivel y secuencia-γ para el problema Levy 3 - Precisión $\epsilon = 10^{-4}$.

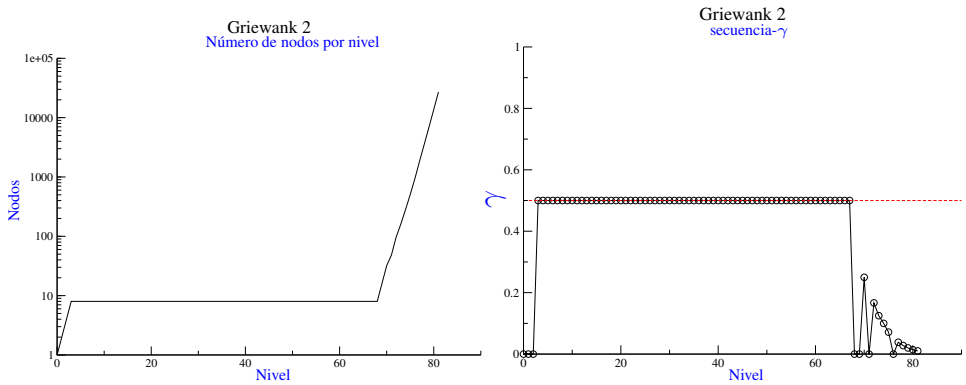

Figura B.3: Número de nodos por nivel y secuencia-γ para el problema Griewank 2 - Precisión $\epsilon = 10^{-9}$.

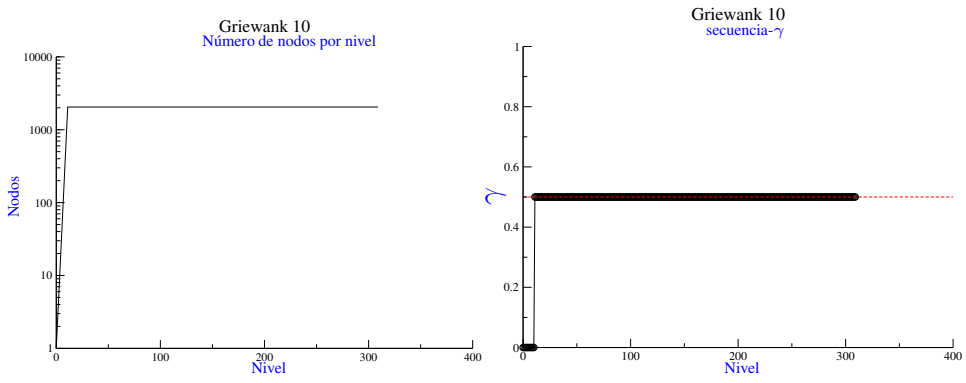

Figura B.4: Número de nodos por nivel y secuencia-γ para el problema Griewank 10 - Precisión $\epsilon = 10^{-6}$.

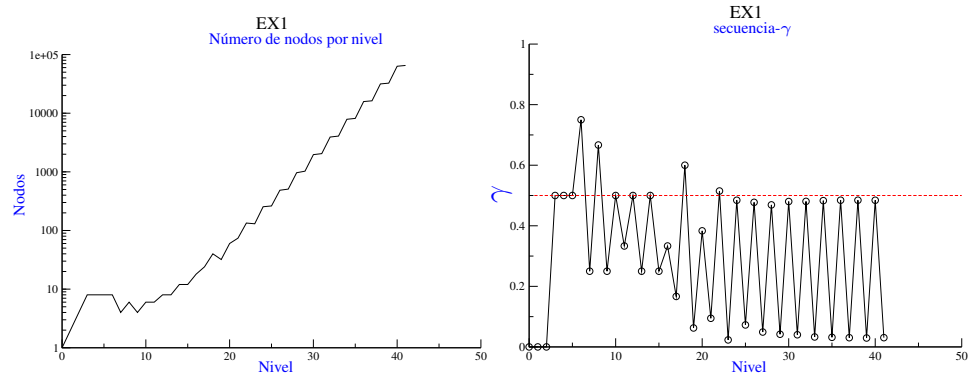

Figura B.5: Número de nodos por nivel y secuencia-γ para el problema EX 1 - Precisión $\epsilon = 10^{-6}$.

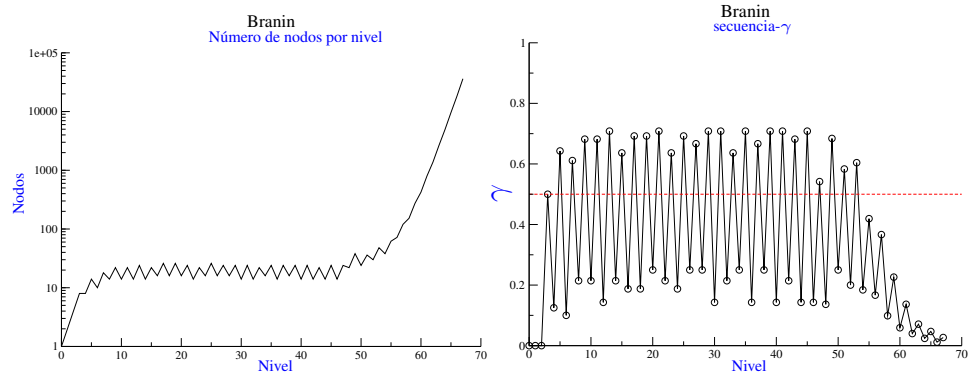

Figura B.6: Número de nodos por nivel y secuencia-γ para el problema Branin - Precisión $\epsilon = 10^{-9}$.

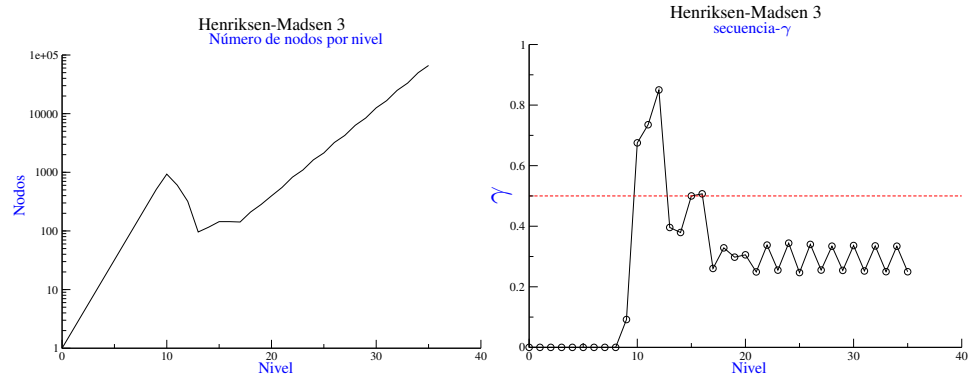

Figura B.7: Número de nodos por nivel y secuencia-γ para el problema Henriksen-Madsen 3 - Precisión $\epsilon = 10^{-4}$.

B. NODOS POR NIVEL Y SECUENCIA-γ DE LAS FUNCIONES DE PRUEBA

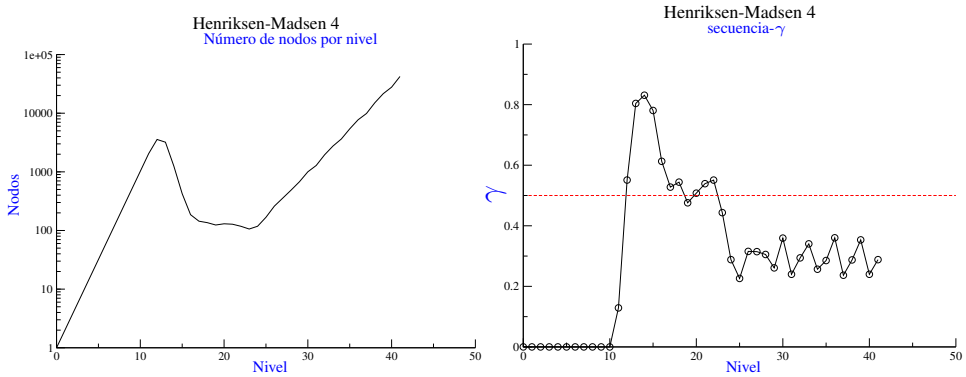

Figura B.8: Número de nodos por nivel y secuencia-γ para el problema Henriksen-Madsen 4 - Precisión $\epsilon = 10^{-3}$.

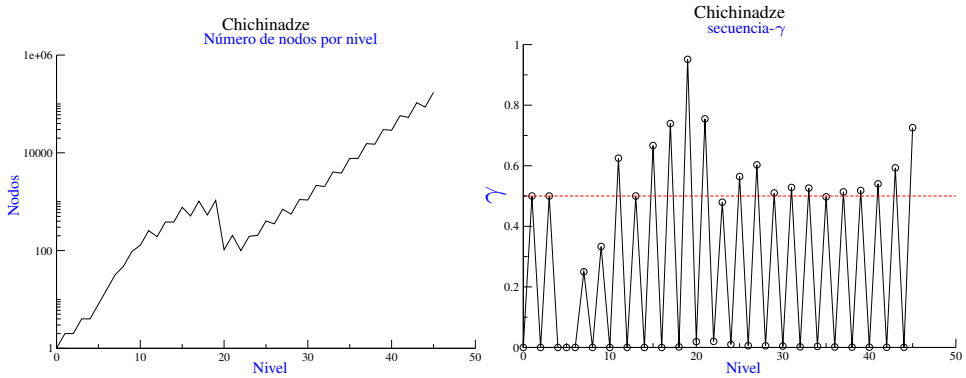

Figura B.9: Número de nodos por nivel y secuencia-γ para el problema Chichinadze - Precisión $\epsilon = 10^{-5}$.

APÉNDICE

C

Figuras de predicción de *left-over*

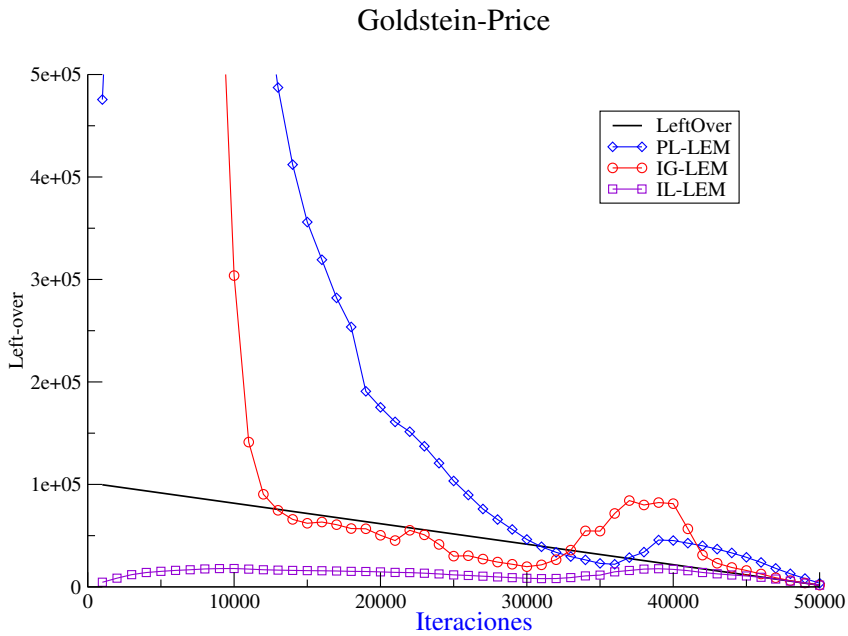

Figura C.1: Valores de predicción del *left-over* para el problema Goldstein Price - Precisión $\epsilon = 10^{-3}$.

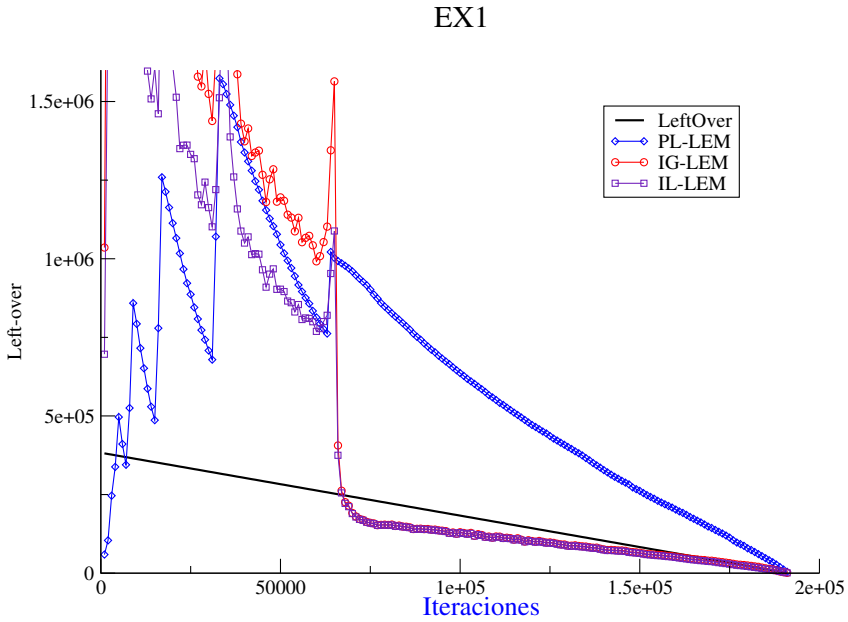

Figura C.2: Valores de predicción del *left-over* para el problema EX 1 - Precisión $\epsilon = 10^{-6}$.

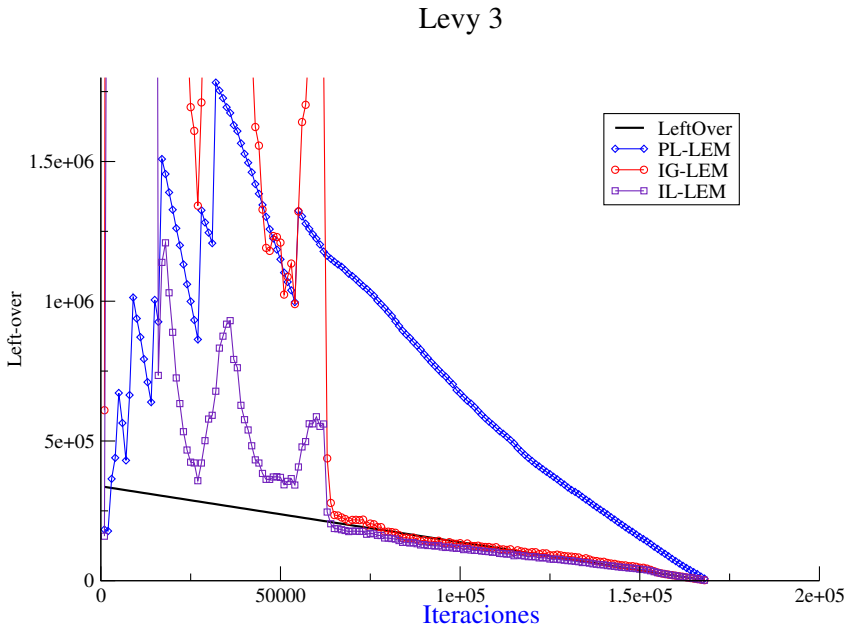

Figura C.3: Valores de predicción del *left-over* para el problema Levy 3 - Precisión $\epsilon = 10^{-4}$.

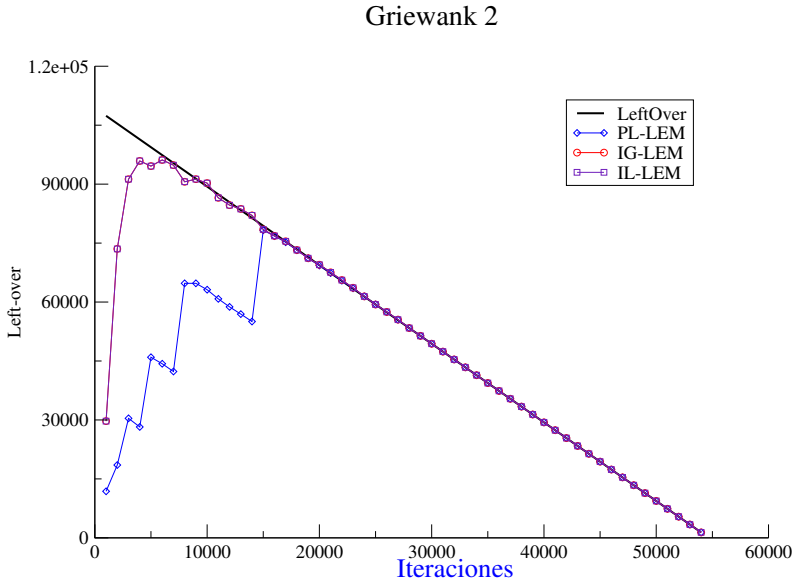

Figura C.4: Valores de predicción del *left-over* para el problema Griewank 2 - Precisión $\epsilon = 10^{-9}$.

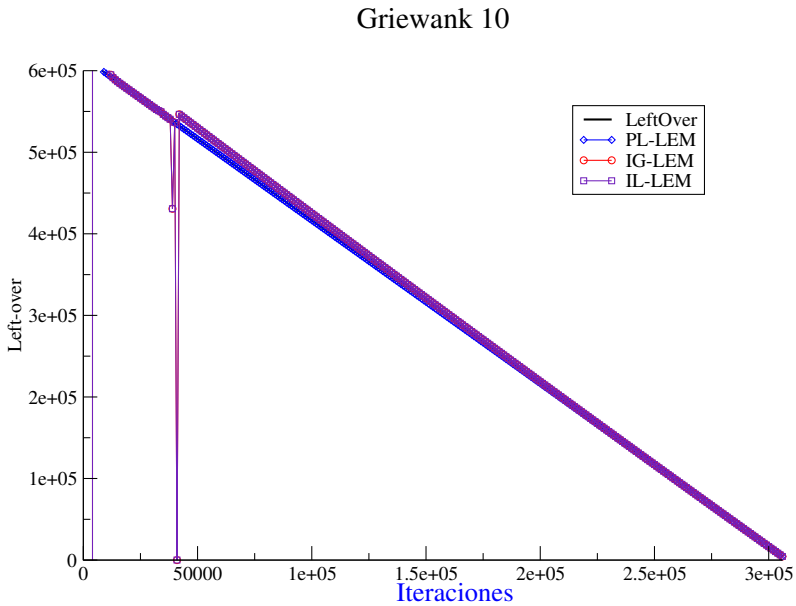

Figura C.5: Valores de predicción del *left-over* para el problema Griewank 10 - Precisión $\epsilon = 10^{-6}$.

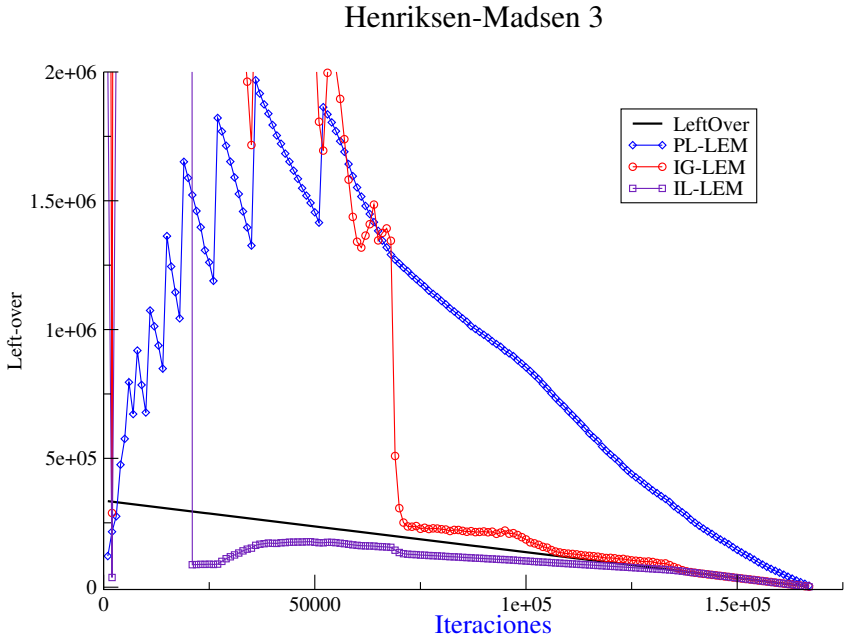

Figura C.6: Valores de predicción del *left-over* para el problema Henriksen-Madsen 3 - Precisión $\epsilon = 10^{-4}$.

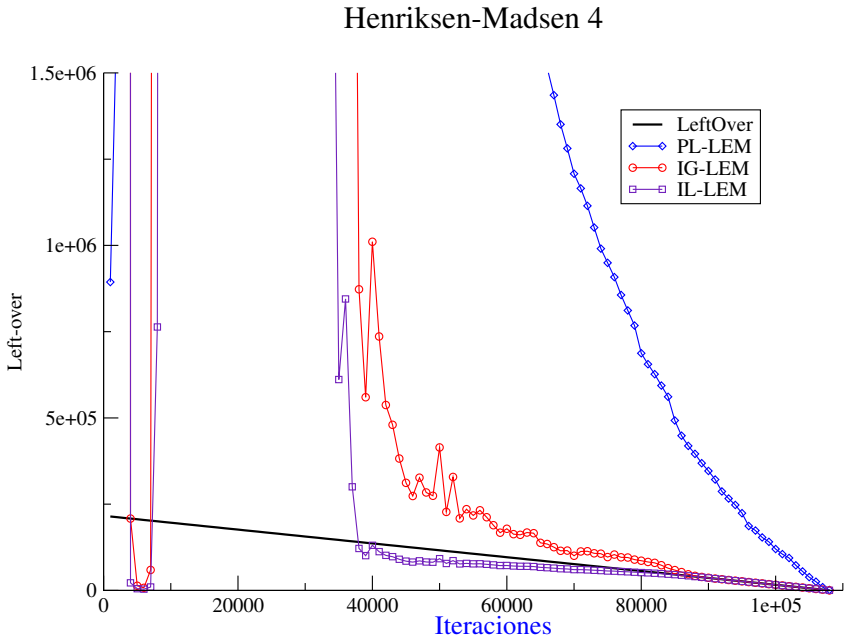

Figura C.7: Valores de predicción del *left-over* para el problema Henriksen-Madsen 4 - Precisión $\epsilon = 10^{-3}$.

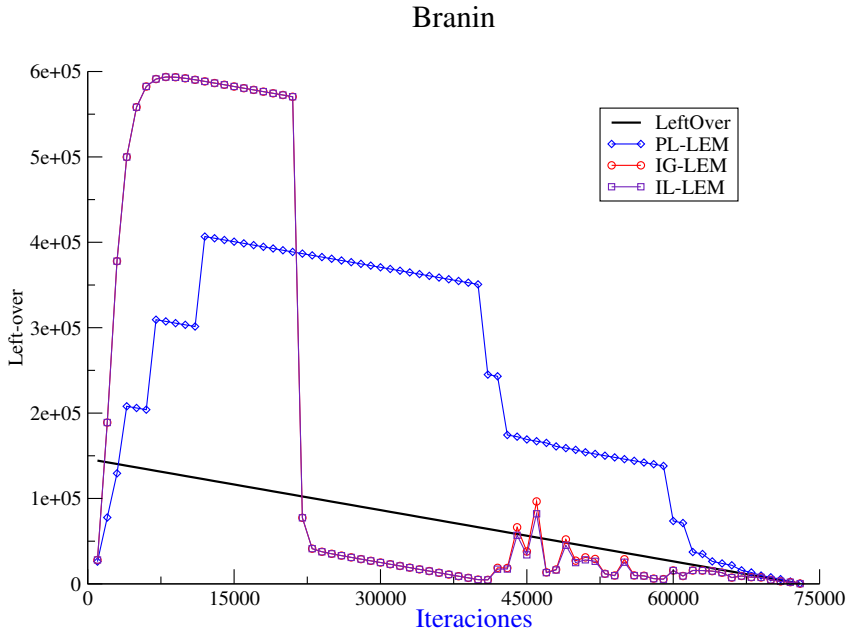

Figura C.8: Valores de predicción del *left-over* para el problema Branin - Precisión $\epsilon = 10^{-9}$.

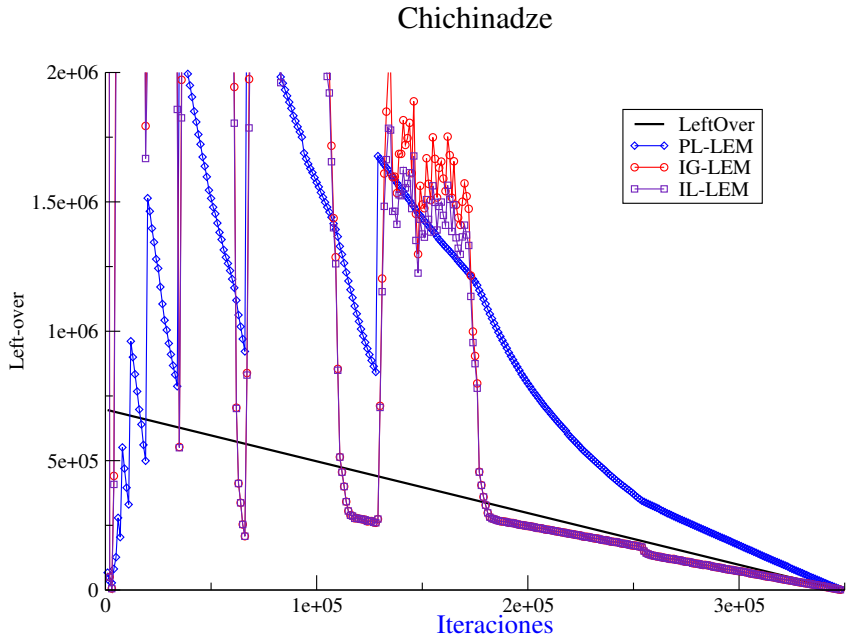

Figura C.9: Valores de predicción del *left-over* para el problema Chichinadze - Precisión $\epsilon = 10^{-5}$.

APÉNDICE

D

Descripción de problemas separables con variable común

Algunas de las siguientes funciones se han elaborado como composición de dos funciones de prueba bien conocidas, en la que se ha compartido una variable común.

1 Example 1. Dimension: 3. $S = [-10, 10]^3$.
$f^{[1]} = x_1^2 + x_1 x_3 + \frac{1}{2}x_3^2 + x_3$, $f^{[2]} = x_2^2 - 2x_2 x_3$
x^* = (5.0, -10.0, -10.0)
f^* = -85.0

2 GP3. Dimension: 3. $S = [-2, 2]^3$.
$f^{[1]}$ = GP(x_1, x_3), $f^{[2]}$ = GP(x_2, x_3)
$x^* \in \{(-0.6, -0.6, -0.4), (-0.4, -0.4, -0.6)\}$
f^* = 65.0

3 L5P. Dimension: 3. $S = [-10, 10]^3$.
$f^{[1]}$ = Levy 5(x_1, x_3), $f^{[2]}$ = Price(x_2, x_3)

x^* = (-1.306853, 0.793737, -1.423764)
f^* = -172.276922

4 L5Pr. Dimension: 3. $S = [-10, 10]^3$.
$f^{[1]}$ = Levy 5(x_1, x_3), $f^{[2]}$ = Price(x_3, x_2)
x^* = (-0.352130, 0.419827, -0.784273)
f^* = -123.743163

5 SHCBL3. Dimension: 3. $S = [-10, 10]^3$.
$f^{[1]}$ = SHCB(x_1, x_3), $f^{[2]}$ = Levy 3(x_3, x_2)
$x^* \in \{$(1.733479, -7.589893, -1.417771),(1.733479, -1.306707, -1.417771), (1.733479, 4.976477, -1.417771)$\}$
f^* = -168.656637

6 G5G5. Dimension: 9. $S = [-600, 600]^9$.
$f^{[1]}$ = G5$(x_1, \ldots, x_4, x_9)$, $f^{[2]}$ = G5$(x_5, \ldots, x_9)$
x^* = (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
f^* = 0.0

7 HPHM4. Dimension: 8. $S = [0.8, 1.2] \times [0.4, 0.8]^2 \times [0.4, 0.6]^5$.
$f^{[1]}$ = HP$(x_1, \ldots, x_5, x_8)$, $f^{[2]}$ = HM4(x_6, x_7, x_8)
x^* = (1.2, 0.8, 0.8, 0.4, 0.57, 0.57, 0.54)
f^* = -9.652140

8 L8HM4. Dimension: 5. $S = [-10, 10]^5$.
$f^{[1]}$ = Levy 8(x_1, x_2, x_5), $f^{[2]}$ = HM4(x_3, x_4, x_5)
$x^* \in \{$ (1, 1, 5.79, 5.79, -0.49), (1, 1, 5.79, -6.77, -0.49), (1, 1, -6.77, 5.79, -0.49), (1, 1, -6.77, -6.77, -0.49), (1, 1, 5.79, -0.49, -0.49), (1, 1, -0.49, 5.79, -0.49), (1, 1, -0.49, -6.77, -0.49), (1, 1, -6.77, -0.49, -0.49), (1, 1, -0.49, -0.49, -0.49)$\}$
f^* = -35.95478

9 RB5G2. Dimension: 6 . $S = [-2, 2]^6$.
$f^{[1]}$ = RBx5$(x_1, \ldots, x_4, x_6)$, $f^{[2]}$ = G2(x_5, x_6)
x^* = (0.97, 0.95, 0.9, 0.81, 0.0, 0.66)
f^* = 0.15487

10 S10S7. Dimension: 7. $S = [0, 10]^7$.

$f^{[1]}$ = S10(x_1, x_2, x_3, x_7), $f^{[2]}$ = S7(x_4, x_5, x_6, x_7)

x^* = (4,4,4,4,4,4,4)

f^* = -20.939349

11 Colville. Dimension: 4. $S = [-10, 10]^4$.

$fa^{[1]}$=$100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1) + 10.1((x_2 - 1)^2 + (x_4 - 1)^2)$

$fa^{[2]}$=$(x_3 - 1)^2 + 90(x_3^2 - x_4)^2$

$fb^{[1]}$=$100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + 10.1(x_2 - 1)^2$

$fb^{[2]}$=$(x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1(x_4 - 1)^2 + 19.8(x_2 - 1)(x_4 - 1)$

x^* = (1,1,1,1)

f^* = 0.0

12 Example 2. Dimension: 5. $S = [-5, 5]^5$.

$f^{[1]}$=$x_5 \sum_{i=1}^{2} x_i \sin x_i^2$

$f^{[2]}$=$x_5 \sum_{i=3}^{5} x_i \sin x_i^2$

$x^* \in \{$(-4.8562, -4.8562, -4.8562, -4.8562, -4.8570), (4.8562, 4.8562, 4.8562, 4.8562, 4.8570)$\}$

f^* = -118.02086

13 Example 3. Dimension: 5. $S = [-5, 5]^5$.

$f^{[1]}$=$\sum_{i=1}^{2}(x_i - 1)^2 - \sum_{i=1}^{2} x_5 x_i$

$f^{[2]}$=$\sum_{i=3}^{5}(x_i - 1)^2 - \sum_{i=1}^{2} x_5 x_i$

x^* = (3.5, 3.5, 3.5, 3.5, 5.0)

f^* = 0.0

A continuación, se describen las funciones que se han utilizado como componentes de las funciones separables presentadas anteriormente:

D. DESCRIPCIÓN DE PROBLEMAS SEPARABLES CON VARIABLE COMÚN

$$
\begin{aligned}
G2(x_1, x_2) &= \frac{x_1^2 + x_2^2}{200} - \cos(x_1) \cdot \cos(\frac{x_2}{\sqrt{2}}) + 1 \\
G5(x_1, x_2, x_3, x_4, x_5) &= \sum_{i=1}^{5} \frac{x_i^2}{4000} - \prod_{i=1}^{5} \cos(\frac{x_i}{\sqrt{2}}) + 1 \\
GP(x_1, x_2) &= [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times \\
&\quad [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \\
HM3(x_1, x_2) &= -\sum_{i=1}^{2} \sum_{j=1}^{5} j \sin((j+1)x_i + j) \\
HM4(x_1, x_2, x_3) &= -\sum_{i=1}^{3} \sum_{j=1}^{5} j \sin((j+1)x_i + j) \\
HP(x_1, x_2, x_3, x_4, x_5, x_6) &= (((x_1 - x_4)^2 + (x_2 - x_5)^2 + (x_3 - x_6)^2)^{-3} - 0.5)^2 \\
Levy\,3(x_1, x_2) &= \sum_{i=1}^{5} i \cos((i+1)x_1 + i) \sum_{j=1}^{5} j \cos((j+1)x_2 + j) \\
Levy\,5(x_1, x_2) &= \sum_{i=1}^{5} i \cos((i+1)x_1 + i) \sum_{j=1}^{5} j \cos((j+1)x_2 + j) \\
&\quad + (x_1 + 1.42513)^2 + (x_2 + 0.80032)^2 \\
Price(x_1, x_2) &= (2x_1^3 x_2 - x_2^3)^2 + (6x_1 - x_2^2 + x_2)^2 \\
RB(x_1.x_2) &= 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 \\
SHCB(x_1, x_2) &= 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \\
Sk(x_1, x_2, x_3, x_4) &= -\sum_{i=1}^{k} \frac{1}{(x - a_i)(x - a_i)^T + c_i}, k = 5, 7, 10
\end{aligned}
$$

con coeficientes:

i	a_i	c_i
1	(4.0,4.0,4.0,4.0)	0.1
2	(1.0,1.0,1.0,1.0)	0.2
3	(8.0,8.0,8.0,8.0)	0.2
4	(6.0,6.0,6.0,6.0)	0.4
5	(3.0,7.0,3.0,7.0)	0.4
6	(2.0,9.0,2.0,9.0)	0.6
7	(5.0,5.0,3.0,3.0)	0.3
8	(8.0,1.0,8.0,1.0)	0.7
9	(6.0,2.0,6.0,2.0)	0.5
10	(7.0,3.6,7.0,3.6)	0.5

Bibliografía

[1] Ahmadi, A., Olshevsky, A., Parrilo, P. & Tsitsiklis, J. (2011). Np-hardness of deciding convexity of quartic polynomials and related problems. *Mathematical Programming*, 1–24.

[2] Alefeld, G. & Herzberger, J. (1983). *Introduction to Interval Computations*. Academic Press, New York.

[3] Applegate, D.L., Bixby, R., Chvátal, V. & Cook, W.J. (1998). On the solution of traveling salesman problems. *Documenta Mathematica*, **Extra Volume Proceedings ICM III**, 645–656.

[4] Applegate, D.L., Bixby, R.E., Chvatal, V. & Cook, W.J. (2007). *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, Princeton, NJ, USA.

[5] Bathia, R. (2007). *Positive definite Matrices*. Princeton University Press, 1st edn.

[6] Baumann, E. (1988). Optimal centered forms. *BIT*, **28**, 80–87.

[7] Bazaraa, M.S., Sherali, H.D. & Shetty, C.M. (1993). *Nonlinear Programming: Theory and Algorithms*. Wiley, New York, 2nd edn.

[8] Berenguel, J.L., Casado, L.G., Hendrix, E.M.T., Messine, F. & García, I. (2010). On interval branch-and-bound algorithm for additively separable functions with one common variable. In *TOGO10: Proceedings of the Toulouse Global Optimization Workshop*, 23–26.

[9] Berenguel, J.L., Casado, L.G., García, I. & Hendrix, E.M.T. (2011). On estimating workload in interval branch-and-bound global optimization algorithms. *Journal of Global Optimization*, 1–24, DOI:10.1007/s10898-011-9771-5.

[10] Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T. & Messine, F. (2012). On lower bounds using additively separable terms in interval B&B. In B. Murgante, O. Gervasi, S. Misra, N. Nedjah, A. Rocha, D. Taniar & B. Apduhan, eds., *Computational Science and Its Applications – ICCSA 2012*, vol. 7335 of *Lecture Notes in Computer Science*, 119–132, Springer Berlin / Heidelberg, DOI:10.1007/978-3-642-31137-6_9.

[11] Berenguel, J.L., Casado, L.G., García, I., Hendrix, E.M.T. & Messine, F. (2012). On interval branch-and-bound for additively separable functions with common variables. *Journal of Global Optimization*, 1–21, DOI:10.1007/s10898-012-9928-x.

[12] Berenguel, J.L., Casado, L.G., Hendrix, E.M.T., Messine, F. & García, I. (2012). On lower bounds using separable terms in interval B&B for one-dimensional problems. In *NAGO12: Proceedings of Global Optimization Workshop 2012*, 39–42.

[13] Berner, S. (1995). *Ein paralleles Verfahren zur verifizierten globalen Optimierung (in German)*. Ph.D. thesis, University of Wuppertal, Germany.

[14] Berner, S. (1996). New results in verified global optimization. *Computing*, **57**, 323–343.

[15] Berner, S. (1996). Parallel methods for verified global optimization practice and theory. *Journal of Global Optimization*, **9**.

[16] Bücker, M., Corliss, G., Hovland, P., Naumann, U. & Norris, B. (2006). *Automatic Differentiation: Applications, Theory, and Implementations (Lecture Notes in Computational Science and Engineering)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[17] Casado, L.G. (1999). *Optimización Global basada en Aritmética de Intervalos y Ramificación y Acotacion: Paralelización*. Ph.D. thesis, Universidad de Malaga, Málaga.

[18] Casado, L.G. & Garcia, I. (1998). New load balancing criterion for parallel interval global optimization algorithms. In *Proceedings of the 16th IASTED International Conference on Applied Informatics, Garmisch-Partenkirchen*, 321–323.

[19] Casado, L.G., García, I. & Csendes, T. (2000). A new multisection technique in interval methods for global optimization. *Computing*, **65**, 263–269.

[20] Casado, L.G., García, I. & Csendes, T. (2001). A heuristic rejection criterion in interval global optimization algorithms. *BIT Numerical Mathematics*, **41**, 683–692, 10.1023/A:1021991817955.

[21] Casado, L.G., Martínez, J.A. & García, I. (2001). Experiments with a new selection criterion in a fast interval optimization algorithm. *Journal of Global Optimization*, **19**, 247–264, 10.1023/A:1011220023072.

[22] Casado, L.G., García, I. & Sergeyev, Y.D. (2002). Interval algorithms for finding the minimal root in a set of multiextremal one-dimensional nondifferentiable functions. *SIAM Journal on Scientific Computing*, **24**, 359–376.

[23] Casado, L.G., García, I., Csendes, T. & Ruíz, V.G. (2003). Heuristic rejection in interval global optimization. *Journal of Optimization Theory and Applications (JOTA)*, **118**, 27–43.

[24] Casado, L.G., García, I., Martínez, J.A. & Sergeyev, Y.D. (2003). New interval analysis support functions using gradient information in a global minimization algorithm. *Journal of Global Optimization*, **25**, 345–362.

[25] Colville, A.R. (1968). A comparative study of nonlinear programming codes. Tech. Rep. 320-2949, IBM Scientific Center, New York.

[26] Comba, J.L.D. & Stolfi, J. (1993). Affine arithmetic and its applications to computer graphics. In *Proceedings of VI SIBGRAPI 1993.*, 9–18, Recife, Brazil, October 1993.

[27] Cornelius, H. & Lohner, R. (1984). Computing the range of values of real functions with accuracy higher than second order. *Computing*, **33**, 331–347, 10.1007/BF02242276.

[28] Cornuéjols, G., Karamanov, M. & Li, Y. (2006). Early estimates of the size of branch-and-bound trees. *INFORMS Journal on Computing*, **18**, 86–96.

[29] Csallner, A.E., Csendes, T. & Csaba Markót, M. (2000). Multisection in interval branch-and-bound methods for global optimization i. theoretical results. *Journal of Global Optimization*, **16**, 371–392.

[30] Csendes, T. (2001). New subinterval selection criteria for interval global optimization. *Journal of Global Optimization*, **19**, 307–327.

[31] Csendes, T. (2004). Generalized subinterval selection criteria for interval global optimization. *Numerical Algorithms*, **37**, 93–100, 10.1023/B:NUMA.0000049489.44154.02.

[32] Csendes, T. (2008). Interval analysis: subdivision directions in interval branch and bound methods. In C.A. Floudas & P.M. Pardalos, eds., *Encyclopedia of Optimization*, 1717–1721, Springer US.

[33] Csendes, T. & Ratz, D. (1997). Subdivision direction selection in interval methods for global optimization. *SIAM Journal on Numerical Analysis*, **34**, 922–938.

[34] de Figueiredo, L.H. & Stolfi, J. (1996). Adaptive enumeration of implicit surfaces with affine arithmetic. *Computer Graphics Forum*, **15**, 287–296.

[35] de Figueiredo, L.H. & Stolfi, J. (1997). *Self-Validated Numerical Methods and Applications*. Brazilian Mathematics Colloquium monographs, IMPA/CNPq, Rio de Janeiro, Brazil.

[36] de Figueiredo, L.H. & Stolfi, J. (2004). Affine arithmetic: Concepts and applications. *Numerical Algorithms*, **37**, 147–158.

[37] Dixon, L.C.W. & Szego, G.P., eds. (1978). *Towards Global Optimization 2*. North-Holland Publishing Company, Amsterdam.

[38] Dwyer, P.S. (1951). Computation with approximate numbers. *Linear Computations*, 11–34.

[39] Eastaman, W.L. (1958). Linear programming with pattern constraints. Tech. Rep. BL 20, The Computation Laboratory, Harvard University, Cambridge.

[40] Ely, J.S. (1990). *Prospects for Using Variable Precision Interval Software in C++ for Solving some Contemporary Scientific Problems*. Ph.D. thesis, Ohio State University.

[41] Eriksson, J. & Lindström, P. (1995). A parallel interval method implementation for global optimization using dynamic load balancing. *Reliable Computing*, **1**.

[42] Ferreira, A. & Pardalos, P., eds. (1996). *Solving Combinatorial Optimization Problems in Parallel*, vol. 1054 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg.

[43] Finkel, R. & Manber, U. (1987). A distributed implementation of backtracking. *ACM Transactions on Programming Languages and Systems*, **9**, 235–256.

[44] Floudas, C.A. & Pardalos, P.M., eds. (2009). *Encyclopedia of Optimization, Second Edition*. Springer.

[45] Garey, M.R. & Johnson, D.S. (1979). *Computer and Intractability*. W. H. Freeman and Company.

[46] Gendron, B. & Crainic, T.G. (1994). Parallel branch-and-bound algorithms: Survey and synthesis. *Operations Research*, **42**, 1042–1066.

[47] Gendron, B. & Crainic, T.G. (1997). A parallel branch-and-bound algorithm for multicommodity location with balancing requirements. *Computers and Operations Research*, **24**, 829–847(19).

[48] Gilmore, P.C. (1962). Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial and Applied Mathematics*, **10**, 305–313.

[49] Gnesi, S., Montanari, U. & Martinelli, A. (1981). Dynamic programming as graph searching: An algebraic approach. *Journal of the ACM*, **28**, 737–751.

[50] Hammer, R. (1995). *C++ Toolbox for Verified Scientific Computing - Theory, Algorithms and Programs: Basic Numerical Problems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[51] Hansen, E.R. (1979). Global optimization using interval analysis – the one dimensional case. *Journal of Optimization Theory and Applications*, **29**, 331–344.

[52] Hansen, E.R. (1980). Global optimization using interval analysis – the multidimensional case. *Numerische Mathematik*, **34**, 247–270.

[53] Hansen, E.R. (1992). *Global Optimization Using Interval Analysis*, vol. 165 of *Pure and Applied Mathematics*. Marcel Dekker, Inc, New York, NY.

[54] Hansen, E.R. & Walster, G.W. (2004). *Global optimization using interval analysis*. Marcel Dekker, 2nd edn.

[55] Hansen, P., Lagouanelle, J.L. & Messine, F. (2007). Comparison between Baumann and admissible simplex forms in interval analysis. *Journal of Global Optimization*, **37**, 215–228.

[56] Hansen, P., Lagouanelle, J.L. & Messine, F. (2007). Comparison between baumann and admissible simplex forms in interval analysis. *Journal of Global Optimization*, **37**, 215–228.

[57] Hendrix, E.M.T. & Tóth, B.G. (2010). *Introduction to Nonlinear and Global Optimization*. Springer Optimization and Its Applications, Springer New York.

[58] Henriksen, T. & Madsen, K. (1992). Parallel algorithms for Global Optimization. *Interval Computations*, **3**.

[59] Henriksen, T. & Madsen, K. (1992). Use of a depth-first strategy in parallel Global Optimization. Tech. Rep. EUR-CS-92-10, Institute for Numerical Analysis, Technical University of Denmark.

[60] Hofschuster, W. & Krämer, W. (2004). C-XSC 2.0 a C++ library for extended scientific computing. *Numerical software with result verification*, 259–276.

[61] Horst, R. & Pardalos, P.M., eds. (1995). *Handbook of Global Optimization*, vol. 2 of *Noncovex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, Holland.

[62] Horst, R., Pardalos, P.M. & Thoai, N.V. (2000). *Introduction to Global Optimization*, vol. 48 of *Non convex optimization and its application (closed)*. Springer-Verlag.

[63] Ibaraki, T. (1976). Theoretical comparisons of search strategies in branch and bound algorithms. *International Journal of Computer and Information Sciences*, **5**, 315–344.

[64] Ibraev, S. (2001). *A New Parallel Method for Verified Global Optimization*. Ph.D. thesis, University of Wuppertal, Wuppertal.

[65] Ichida, K. & Fujii, Y. (1979). An interval arithmetic method for global optimization. *Computing*, 85–97.

[66] IEEE Task P754 (2008). *IEEE 754-2008, Standard for Floating-Point Arithmetic*. IEEE, New York, NY, USA.

[67] Kahan, W.M. (1968). A more complete interval analysis. In *In lecture Notes for a Summer Course at the University of Michigan*.

[68] Kahou, J.H.T. (2005). *Some new Acceleration Mechanisms in Verified Global Optimization*. Ph.D. thesis, Bergischen Unviersität Wuppertal.

[69] Kearfott, R.B. (1996). *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht, Holland.

[70] Kearfott, R.B. (1997). Empirical evaluation of innovations in interval branch and bound algorithms for nonlinear systems. *SIAM Journal on Scientific Computing*, **18**, 574–594.

[71] Keesman, K. (1992). Determination of a minimum-volume orthotopic enclosure of a finite vector set. Tech. Rep. MRS Report 92-01, Wageningen Agricultural University.

[72] Khachiyan, L. & Todd, M.J. (1993). On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematical Programming*, **61**, 137–159.

[73] Kilby, P., Slaney, J., Thiébaux, S. & Walsh, T. (2006). Estimating search tree size. In *AAAI'06: proceedings of the 21st national conference on Artificial intelligence*, 1014–1019, AAAI Press.

[74] Knuth, D. (1975). Estimating the efficiency of backtrack programs. *Mathematics of Computation*, **29**, 121–136.

[75] Krawczyk, R. & Nickel, K. (1982). The centered form in interval arithmetic: quadratic convergence and inclusion isotonicity. *Computing*, **28**, 117–137.

[76] Kreinovich, V. & Csendes, T. (2001). Theoretical justification of a heuristic subbox selection criterion for interval global optimization. *Central European Journal of Operations Research*, **9**, 255–265.

[77] Kubica, B.J. (2012). A class of problems that can be solved using interval algorithms. *Computing*, **94**, 271–280.

[78] Kumar, V. & Kanal, L. (1983). The CPD: a unifying formulation for heuristic search, dynamic programming and branch and bound. In *National Conference on Artificial Intelligence*.

[79] Lagouanelle, J.L. & Soubry, G. (2004). Optimal multisections in interval branch-and-bound methods of global optimization. *Journal of Global Optimization*, **30**, 23–38, 10.1023/B:JOGO.0000049095.55259.61.

[80] Land, A.H. & Doig, A.G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, **28**, 497–520.

[81] Laursen, P.S. (1993). Simple approaches to parallel branch and bound. *Parallel Computing*, **19**, 143–152.

[82] Lawler, E.L. & Wood, D.E. (1966). Branch and bound methods: A survey. *Operation Research*, **14**, 699–719.

[83] Leclerc, A.P. (1992). *Efficient and Reliable Global Optimization*. Ph.D. thesis, Ohio State University.

[84] Little, J.D.C., Murty, K.G., Sweeney, D.W. & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research*, **11**, 972–989.

[85] Loh, E. & Walster, G.W. (2002). Rump's example revisited. *Reliable Computing*, **8**, 245–248, 10.1023/A:1015569431383.

[86] Markov, S. (1980). Some applications on extended interval arithmetic to interval iterations. *Computing (Suppl.)*, **2**, 69–84.

[87] Markov, S. (1981). On interval arithmetic and its applications. In *Proceedings of the 5th Symposium on computer Arithmetic*, IEEE, U. Michigan, MI, USA.

[88] Markót, M.C., Csendes, T. & Csallner, A.E. (2000). Multisection in interval branch-and-bound methods for global optimization ii. numerical tests. *Journal of Global Optimization*, **16**, 219–228.

[89] Markót, M.C., Fernandez, J., Casado, L.G. & Csendes, T. (2006). New interval methods for constrained global optimization. *Mathematical Programming Series A*, **106**, 287–318.

[90] Martínez, J.A., Casado, L.G., García, I., Sergeyev, Y.D. & Tóth, B.G. (2004). On an efficient use of gradient information for accelerating interval global optimization algorithms. *Numerical Algorithms*, **37**, 61–69.

[91] Messine, F. (2002). Extentions of affine arithmetic: Application to unconstrained global optimization. *Journal of Universal Computer Science*, **8**, 992–1015.

[92] Messine, F. & Lagouanelle, J.L. (1998). Enclosure methods for multivariate differentiable functions and application to global optimization. *Journal of Universal Computer Science*, **4**, 589–603.

[93] Messine, F. & Nogarede, B. (2006). Optimal design of multi-airgap electrical machines: An unknown size mixed-constrained global optimization formulation. *IEEE Transactions on Magnetics*, **42**, 3847–3853.

[94] Messine, F. & Touhami, A. (2006). A general reliable quadratic form: An extension of affine arithmetic. *Reliable Computing*, **12**, 171–192.

[95] Mitten, L.G. (1970). Branch and bound methods: general formulation and properties. *Operation Research*, **18**, 24–34.

[96] Moore, R.E. (1962). *Interval arithmetic and automatic error analysis in digital computing*. Ph.D. dissertation, Department of Mathematics, Stanford University, Stanford, CA, USA, also published as Applied Mathematics and Statistics Laboratories Technical Report No. 25.

[97] Moore, R.E. (1966). *Interval analysis*. Prentice-Hall, New Jersey, (USA).

[98] Moore, R.E. (1976). On computing the range of a rotational function of n variables over a bounded region. *Computing*, **16**, 1–15.

[99] Moore, R.E. (1977). A test for existence of solutions to nonlinear systems. *SIAM Journal on Numerical Analysis*, **14**, 611–615.

[100] Moore, R.E. & Jones, S.T. (1977). Safe starting regions for iterative methods. *SIAM Journal on Numerical Analysis*, **14**, 1051–1065.

[101] Moore, R.E., Hansen, E.R. & Leclerc, A.P. (1992). Rigorous methods for parallel Global Optimization. In C. Floudas & P. Pardalos, eds., *Recent Advances in Global Optimization*, Princenton University Press.

[102] Moore, R.E., Kearfott, R.B. & Cloud, M.J. (2009). *Introduction to Interval Analysis*. SIAM, Philadelphia, (USA).

[103] Morin, T. & Marsten, R. (1976). Branch-and-bound strategies for dynamic programming. *Operations Research*, **24**, 611–627.

[104] Nau, D.S., Kumar, V. & Kanal, L.N. (1984). General branch and bound and its relation to A* and AO*. *Artificial Intelligence*, **23**, 29–58.

[105] Neumaier, A. (1990). *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge.

[106] Neumaier, A. (2004). Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, **13**, 271–369.

[107] Nocedal, J. & Wright, S.J. (1999). *Numerical Optimization*. Springer, New York.

[108] Novoa, M. (1993). Advances in the interval solution of algebraic systems.

[109] Özaltın, O.Y., Hunsaker, B. & Schaefer, A.J. (2011). Predicting the solution time of branch-and-bound algorithms for mixed-integer programs. *INFORMS Journal on Computing*, **23**, 392–403.

[110] Pardalos, P.M., ed. (1999). *Parallel processing of discrete problems*, vol. 106 of *The IMA volumes in mathematics and its applications*. Springer-Verlag.

[111] Pardalos, P.M. & Schnitger, G. (1988). Checking local optimality in constrained quadratic programming is NP-hard. *Operations Research Letters*, **7**, 33–35.

[112] Pardalos, P.M. & Vavasis, S.A. (1991). Quadratic programation with one negative eigenvalue is NP-hard. *Journal of Global Optimization*, **1**, 15–22.

[113] Pardalos, P.M. & Vavasis, S.A. (1992). Open questions in complexity theory for numerical optimization. *Mathematical Programming*, **57**, 337–339, 10.1007/BF01581088.

[114] Pardalos, P.M., Resende, M.G.C. & Ramakrishnan, K.G., eds. (1995). *Parallel processing of discrete optimization problems: DIMACS workshop, April 28-29, 1994*, vol. 22 of *DIMACS series in discrete mathematics and theoretical computer science*. American Mathematical Society.

[115] Pintér, J.D. (1996). Continuous global optimization software: A brief review. *Newsletter of the Mathematical Programming Society*, **52**, 1–8.

[116] Rao, V.N. & Kumar, V. (1987). Parallel depth first search. part I. Implementation. *International Journal of Parallel Programing*, **16**, 479–499.

[117] Ratschek, H. & Rokne, J. (1984). *Computer methods for the range of functions*. Ellis Horwood Ltd, Chichester.

[118] Ratschek, H. & Rokne, J. (1988). *New Computer Methods for Global Optimization*. Ellis Horwood Ltd., Chichester, England.

[119] Ratz, D. (1992). *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. Ph.D. thesis, University of Karlsruhe.

[120] Ratz, D. (1995). On branching rules in second-order branch-and-bound methods for global optimization. *Scientific Computing and Validated Numerics*, 221–227.

[121] Ratz, D. (1996). On extended interval arithmetic and inclusion isotonicity. institut für angewandte mathematik. Preprint, U. Karlsruhe, Germany.

[122] Ratz, D. (1998). Automatic slope computation and its application in nonsmooth global optimization. *Journal of Global Optimization*, 393.

[123] Ratz, D. (1999). A nonsmooth global optimization technique using slopes: The one-dimensional case. *Journal of Global Optimization*, **14**, 365–393.

[124] Ratz, D. & Csendes, T. (1995). On the selection of subdivision directions in interval branch and bound methods for global optimization. *Journal of Global Optimization*, **7**, 183–207.

[125] Rossman, M.J., Twery, R.J. & Stone, F.D. (1958). A solution to the traveling salesman problem by combinatorial programming. Chicago.

[126] Rump, S.M. (1988). Algorithm for verified inclusions – theory and practice. In R. Moore, ed., *Reliability in Computing*, 109–126, Academic Press.

[127] Schoen, F. (1991). Stochastic techniques for global optimization: A survey of recent advances. *Journal of Global Optimization*, **1**, 207–228, 10.1007/BF00119932.

[128] Skelboe, S. (1974). Computation of rational interval functions. *Bit Numerical Mathematics*, **14**, 87–95.

[129] Sunaga, T. (1958). Theory of interval algebra and its application to numerical analysis. *Research Association of Applied Geometry (RAAG) Memoirs*, **3**, 29–46.

[130] Törn, A. & Žilinskas, A. (1989). *Global Optimization*, vol. 3350. Springer-Verlag, Berlin, Germany.

[131] Tóth, B. & Casado, L.G. (2007). Multi-dimensional pruning from baumann point in an interval global optimization algorithm. *Journal of Global Optimization*, **38**, 215–236.

[132] Tóth, B. & Csendes, T. (2005). Empirical investigation of the convergence speed of inclusion functions in a global optimization context. *Reliable Computing*, **11**, 253–273.

[133] Trienekens, H.W.J.M. & De Bruin, A. (1992). Towards a taxonomy of parallel branch and bound algorithms. Tech. Rep. EUR-CS-92-01, Erasmus University Rotterdam.

[134] Vinkó, T., Lagouanelle, J.L. & Csendes, T. (2004). A new inclusion function for optimization: Kite – the one dimensional case. *Journal of Global Optimization*, **30**, 435–456.

[135] Wiethoff, A. (1998). *Verifizierte Globale Optimierung auf Parallelrechnern*. Ph.D. thesis, Universität Karlsruhe, Karlsruhe.

Declaración

Declaro que he producido este trabajo sin la asistencia de terceras partes y sin hacer uso de otras ayudas distintas a las mecionadas. Los datos tomados directa o indirectamente de otras fuentes han sido adecuadamente identificadas. Por último, este trabajo no ha sido utilizado previamente de forma idéntica o similar en otro tribunal de tesis doctoral.

La tesis doctoral fue dirigida desde 2007 hasta 2012 bajo la supervisión de Leocadio González Casado y Eligius M.T. Hendrix en la Universidad de Almería.

Almería,

La tesis doctoral fue finaliza en Almería el 9 de octubre de 2012

Esta página se ha dejado en blanco intencionadamente