Departamento de Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingeniería Informática
UNIVERSIDAD DE GRANADA

NUEVOS CRITERIOS DE PARADA EN ALGORITMOS DE OPTIMIZACIÓN

TESIS DOCTORAL

Edmundo Rubén Vergara Moreno

Granada - España
1999

NUEVOS CRITERIOS DE PARADA EN ALGORITMOS DE OPTIMIZACIÓN

MEMORIA QUE PRESENTA

Edmundo Rubén Vergara Moreno

PARA OPTAR AL GRADO DE DOCTOR EN MATEMÁTICAS

Mayo de 1999

DIRECTOR

José Luis Verdegay Galdeano

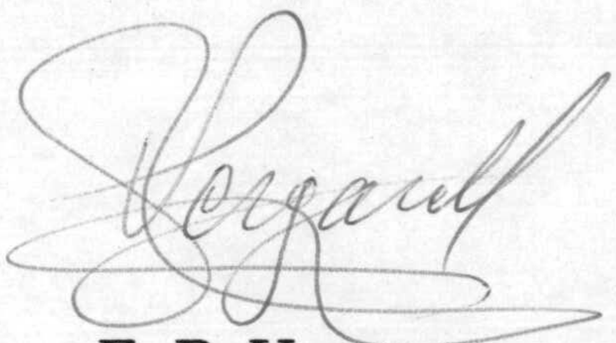
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIAL
E.T.S. de INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE GRANADA

ESPAÑA

La memoria titulada **Nuevos Criterios de Parada en Algoritmos De Optimización,** que presenta D. Edmundo Rubén Vergara Moreno para optar al grado de Doctor, ha sido realizada en el departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada bajo la dirección de D. José Luis Verdegay Galdeano.

Granada, Mayo de 1999

El Doctorando



E. R. Vergara

El Director

J. L. Verdegay

AGRADECIMIENTOS

En estos momentos en que termino de escribir la presente Memoria, invaden mi mente los nombres de todas las personas e instituciones que de una u otra forma me han facilitado alcanzar este anhelo, y a quienes estoy muy agradecido. Sin embargo quiero hacer expreso mi agradecimiento a aquellos que en forma directa me han brindado su ayuda.

Mi gratitud al Gobierno Español, quien a través de la Agencia Española de Cooperación Internacional, me otorgó una Beca ICI por el período de tres cursos. Asimismo, a la Facultad de Ciencias Físicas y Matemáticas de la Universidad Nacional de Trujillo-Perú, por haberme concedido la licencia correspondiente, y al Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada por su acogida y por haberme brindado todo el apoyo para llevar a cabo esta Memoria.

Mis agradecimientos al Dr. José Luis Verdegay Galdeano por la confianza que depositó en mí, prestándome su asesoramiento y dirección, que durante este tiempo se vio reflejado en sus orientaciones expertas y supervisión constante.

Mis agradecimientos al Dr. Salomón Espinoza Quiroz por su apoyo constante y al Dr. Pedro Lavalle, quienes me dieron amplio apoyo para hacer posible mi viaje a España.

NUEVOS CRITERIOS DE PARADA EN ALGORITMOS DE OPTIMIZACIÓN

Edmundo Rubén Vergara Moreno

ÍNDICE

INTRODUCCIÓN

El hombre es el único ser sobre la faz de la tierra que ha luchado y lucha no sólo por su supervivencia sino para que su estancia sea cada día más placentera. Aunque muchos pasajes de la historia de la humanidad se han caracterizado por la lucha para lograr vivir mejor el presente y lograr la complacencia individual propia, la lucha incesante y de mayor predominancia ha sido por lograr el bienestar futuro y de la sociedad en general.

Desde las épocas más remotas en que inicialmente los medios de supervivencia eran la recolección, la caza y la pesca, el hombre no sólo se ha limitado a consumir lo que la naturaleza le ha brindado sino que ha utilizado su inteligencia para aprovecharla de la mejor manera. Al usar su inteligencia fabricó y fabrica aún hoy las herramientas para mejorar sus capacidades, descubrió la agricultura para no esperar que la tierra produjera de manera accidental, descubrió y sigue descubriendo las leyes naturales para luego usar aquellas que son beneficiosas o bien para evitarlas si son perjudiciales. Todo ello es lo que ahora se llama ciencia y tecnología.

Como producto de la ciencia y la tecnología, hoy en día, la humanidad cuenta con unas poderosas herramientas y entre ellas están los ordenadores. Los ordenadores son herramientas con capacidades programables ilimitadas en su esencia. Las limitaciones se dan en la programación y en los accesorios necesarios para una determinada tarea. Pueden ejecutar tareas muy simples y repetitivas, tareas arriesgadas u operaciones matemáticas materialmente imposibles para el hombre e incluso actividades de razonamiento propias de la mente humana.

Actualmente debido a esas capacidades, los ordenadores constituyen instrumentos de ayuda imprescindibles para los gestores de diversos tipos de instituciones industriales, comerciales, de servicios, etc. en la solución de diversos problemas.

Al hablar de las ayudas que proporcionan los ordenadores a los gestores no nos referimos a la ayuda como simples almacenes de documentos personales importantes

sino como sistemas complejos que involucran diversas acciones concertadas de órganos con gran impacto potencial en la efectividad de las decisiones gerenciales.

El desarrollo e incluso la supervivencia o la quiebra de las grandes organizaciones públicas y privadas dependen de las decisiones que deben tomar sus administradores; decisiones que en muchos casos deben ser tomadas de una manera rápida e inteligente.

Durante muchos años se ha considerado que la toma de decisión era un arte, de tal manera que se podía usar una gran variedad de estilos individuales en la solución aproximada de problemas gerenciales similares. Estos estilos han estado basados frecuentemente en la creatividad, el juicio, la intuición y la experiencia más que en métodos sistemáticos cuantitativos o en aproximaciones científicas.

Sin embargo en la actualidad, en el ambiente rápidamente cambiante y cada vez más complejo en que vivimos, la toma de decisiones se hace más difícil por dos razones: primero, debido a las tecnologías y los sistemas de comunicación altamente desarrollados, el número de alternativas disponibles es muy elevado, y segundo, porque el costo de los errores cometidos puede ser muy grande como consecuencia de la complejidad y la magnitud de operaciones, la automatización y la reacción en cadena que un error puede generar en muchas partes de una organización. Por el contrario, los beneficios pueden ser extremadamente grandes cuando las decisiones tomadas son acertadas.

En un ambiente caracterizado por factores tecnológicos en constante crecimiento, dentro de una estructura compleja, y en un mercado internacional de competición, la toma de decisiones se hace cada vez más sofisticada y por tanto requiere del uso de las tecnologías que se están desarrollando. Más aún, se requieren mejores tecnologías como herramientas de ayuda en la toma de decisiones.

Las tecnologías de sistemas de ayuda a esa toma de decisiones que se están desarrollando [197] son: Sistemas Soporte de Decisión (SSD), Sistemas Soporte de Decisión en Grupo (SSDG), Sistemas de Información Ejecutiva (SIE), Sistemas Expertos (SE), Redes Neuronales Artificiales (RNA) y Sistemas Soporte Híbridos (SSH). La aplicación de estas tecnologías se denomina en conjunto "sistemas de ayuda en la administración".

Según Simon [180], el proceso de la toma de decisión, en forma general, se realiza en un rango continuo de decisiones que varía entre las estructuradas y las no estructuradas, entendiéndose por una situación estructurada o llamada también

programada aquella que tiene un contexto con elementos y relaciones entre ellos que son o pueden ser completamente conocidos o determinados.

Según Gorry y Scott-Morton [76], para la toma de decisiones en un ambiente estructurado es suficiente hacer uso de SIE, de modelos de investigación de operaciones, etc. porque el carácter estructurado sólo está presente en administraciones de bajo nivel, mientras que en el caso de situaciones semi-estructuradas o no estructuradas es necesario el uso de SSD, SSDG, RNA o SSH, de acuerdo al caso; estas situaciones son propias del nivel funcional de los altos ejecutivos y profesionales altamente especializados en problemas complejos.

De entre los sistemas de ayuda a la toma de decisiones citados más arriba, destacan por su importancia los SSD. El concepto involucrado en los SSD fue articulado por primera vez en 1971 por Scott-Morton, quien los entendió como sistemas interactivos basados en computación que ayudan al decisor a utilizar los datos y los modelos para resolver problemas no estructurados. Otra definición de Keen y Scott-Morton [101] dice que los SSD unen los recursos intelectuales de los individuos con las capacidades de los ordenadores para mejorar la calidad de las decisiones en problemas semiestructurados o no estructurados.

A partir de éstas y otras definiciones podemos resumir que un SSD es un sistema de ayuda en la toma de decisiones en situaciones en el que los datos y/o sus relaciones no son conocidos exactamente, o no pueden ser determinados con exactitud. Los componentes básicos de un SSD [197] son el Sistema de Gestión de la Base de Datos (SGBD), el Sistema de Gestión de la Base de Modelos (SGBM) y el Sistema de Gestión y Generación de Diálogos (SGGD).

Un buen SSD será aquél en que cada uno de sus componentes tenga la mejor capacidad posible y al mismo tiempo estén bien integrados entre sí. Decir que los componentes de un SSD tengan la mejor capacidad posible significa que el SGBD debe tener la capacidad de tratar una gran variedad de estructuras de datos que permitan el manejo de datos probabilísticos, incompletos e imprecisos; que el SGBM sea capaz de simular diversas situaciones reales y finalmente, que el SGGD tenga la suficiente capacidad de admitir, transmitir y reportar todo tipo de información entre el usuario y el SSD.

Un buen SGBM debe proporcionar la capacidad de implementar diversas posibilidades de realizar análisis sofisticados y diversas capacidades de interpretación;

igualmente debe habilitar al decisor para explorar diversas situaciones de decisión a través del uso de la base de datos mediante una base de modelos de procedimientos algorítmicos y protocolos asociados a la gestión de modelos. Esto puede hacerse usando sentencias de modelización, en algún lenguaje procedimental o no procedimental, mediante el uso de modelos subrutinas, tales como los paquetes de programación matemática.

Es deseable la existencia de diversos modelos para que se puedan acomodar mejor y más flexiblemente a los deseos del decisor. Naturalmente los diversos modelos en SGBM pre-escritos deben ser aquéllos cuya utilidad haya sido demostrada con anterioridad, tales como los de la programación lineal, los modelos de análisis de decisiones multiatributo. Asimismo, deben ser bien especificados los procedimientos para el uso de estos modelos. El SGBM también debería permitir al usuario construir sus propios modelos, así como heurísticas a partir de modelos pre-establecidos. Igualmente, debe posibilitar la realización de análisis de sensibilidad sobre los resultados que se obtengan y la ejecución de modelos con una amplia gama de datos de cara a poder contestar preguntas del tipo "¿qué pasa si...?".

El propósito de un SGBM es fundamentalmente transformar los datos del SGBD en información que sea útil para el decisor. Esta tarea la realiza mediante la creación, almacenamiento, acceso y manipulación de modelos.

Los modelos no son más que las representaciones de los datos y sus relaciones. El proceso de construcción de dicha representación se denomina modelización. El proceso de modelización y por tanto, los modelos que representan a un conjunto de datos y sus relaciones dependen de los objetivos y de la naturaleza de la institución que lo utiliza, lo cual se debe a que los datos y sus interrelaciones no tienen una única representación.

En los casos de asignación de recursos bajo restricciones, de planificación, de análisis de gestión, etc. los modelos utilizados se denominan *modelos de programación matemática*, mientras que para determinar el mejor control o acción cuando el sistema, los controles, las restricciones o los índices de calidad (evaluación) pueden cambiar con el tiempo, se utilizan *modelos denominados de control óptimo de sistemas*; por citar sólo algunos tipos.

La asignación de recursos limitados o bajo restricción y el control de acciones cambiantes con el tiempo son situaciones que están presentes en toda institución pequeña o grande, pública o privada. Son situaciones problemáticas en las cuales los

modelos y sus respectivos métodos de solución existentes dejan un campo abierto por modelar y donde plantear los correspondientes métodos de solución. Porque los modelos y métodos conocidos o bien no representan a la realidad o bien sus métodos de solución presentan limitaciones muy serias.

De esta amplitud de problemas dentro de la asignación de recursos limitados y del control de acciones cambiantes con el tiempo, vamos a prestar atención sólo a dos situaciones:

1) cuando los datos no se conocen exactamente, es decir, cuando los datos son imprecisos y se está en un ambiente semi-estructurado, y
2) cuando los métodos de solución existentes que sirven para casos simples presentan limitaciones o simplemente no se pueden utilizar en casos complejos.

Con la introducción de *la teoría de conjuntos difusos* en los modelos y métodos de la programación matemática se está abordando con cierto éxito el primer caso. Una de las partes más importantes de esta integración es aquélla que corresponde a la integración de la programación lineal clásica con la teoría de los conjuntos difusos que ha dado origen a la denominada **Programación Lineal Difusa**, la cual desde su aparición en los comienzos de los años 70 no ha cesado en su desarrollo hasta la actualidad.

En el segundo caso están incluidos aquellos problemas en los cuales el uso de métodos exactos de solución de los modelos existentes tiene limitaciones en el tiempo de ejecución, cuando se utilizan dichos métodos en situaciones con muchas variables de decisión y muchas restricciones. Para estos problemas se han desarrollado diversos métodos heurísticos, tales como los algoritmos genéticos, el algoritmo recocido simulado, las redes neuronales artificiales, etc. que proporcionan soluciones aproximadas y no exactas.

Estos dos aspectos del SGBM de un SSD, que son de trascendental importancia en la actualidad, han captado nuestra atención y han constituido el motor de desarrollo del presente trabajo cuya presentación hacemos en tres capítulos.

En el primer capítulo, se aborda una revisión de la Programación Lineal Difusa, en la que se recogen los principales modelos y sus respectivos métodos de solución. Se detallan dos o tres de los más importantes métodos y se hace un comentario de otros.

Luego revisamos, también de la misma manera, las extensiones y aplicaciones más destacadas realizadas hasta la actualidad.

El otro aspecto de nuestro interés se desarrolla en los dos capítulos restantes. En ellos introducimos los *nuevos criterios de parada* basados en reglas difusas en algoritmos exactos, tradicionalmente muy importantes. Dichos criterios de parada transforman a los algoritmos exactos en algoritmos de aproximación, enriqueciendo consecuentemente, la base de algoritmos aproximados o heurísticas para encarar problemas de grandes dimensiones o de naturaleza compleja.

Como se sabe, los algoritmos de aproximación o heurísticas tienen lugar en problemas denominados NP-completos, en aquellas situaciones donde los algoritmos exactos requieren tiempos de ejecución muy elevados, que en algunas casos pueden rebasar con creces el tiempo de vida del hombre. Esto ocurre, por ejemplo con el algoritmo simplex de la programación lineal, que es muy eficiente para algunas situaciones y no lo es en general. Este algoritmo y otros algoritmos más eficientes de la programación lineal pueden ser mejorados en su tiempo de ejecución, aunque a costa de su exactitud, con la introducción de criterios de parada difusos.

Los problemas, por excelencia, NP-completos son el problema de la mochila y el problema del viajante de comercio, que carecen de algoritmos con tiempos de ejecución polinomial; los algoritmos exactos que existen sólo son aplicables para dimensiones pequeñas, y son los algoritmos de aproximación o heurísticas los que se utilizan en grandes magnitudes. En este trabajo hemos introducido, en aquellos algoritmso exactos, los criterios de parada difusos de tal manera que se tengan nuevos algoritmos de aproximación. Todo esto se aborda en en esta memoria del siguiente modo:

En el capítulo 2, se hace una revisión muy resumida de conceptos asociados con algoritmos, resaltando los criterios de parada que los caracterizan, y luego se introduce el concepto de criterio de parada difuso. Inmediatamente, se prosigue revisando la introducción de criterios de parada difusos en el algoritmo simplex, luego nos centramos en la introducción de criterios de parada difusos en el algoritmo de Karmarkar, en alguna de sus extensiones inmediatas y finalmente introducimos los criterios de parada difusos en los algoritmos de puntos interiores, todos estos dentro de la programación lineal.

En el tercer capítulo se hace la introducción de los criterios de parada difusos en los problemas de la mochila y del viajante de comercio. En ambos problemas se diseñan reglas de parada para los algoritmos de ramificación y acotación, y para los algoritmos de la programación dinámica.

Finalizamos la memoria con la exposición de las conclusiones mas importantes y una relación de la bibliografía consultada para su elaboración.

Capítulo 1

PROGRAMACIÓN LINEAL DIFUSA

1.1 INTRODUCCIÓN

Dentro de una economía del bienestar, existe un problema simple pero de mucha importancia cuando se tienen unas cantidades dadas de varios factores de producción y cierto número de tareas a las que aquéllos se pueden destinar. Estos factores de producción pueden ser asignados a las diferentes tareas, por lo general de muchas maneras distintas, con resultados diversos.

Debido a que la representación de las restricciones toma la forma más sencilla de las funciones matemáticas, como son las funciones lineales, los problemas anteriores se denominan *problemas de la economía lineal.* Dentro de esta denominación están comprendidos no solamente problemas del tipo de los citados arriba sino también muchos otros problemas de la economía.

Las particularidades que presenta la mejor asignación de factores en casos como éste han sido estudiadas desde finales del siglo XIX. Pero los métodos importantes y válidos, incluso hasta la actualidad, se han desarrollado en la primera mitad del presente siglo XX. Podemos destacar tres métodos que nacieron separadamente, aunque finalmente se unieron en uno solo. El primero en aparecer es la denominada teoría de juegos, iniciada por John Von Neumann en 1928, y luego aplicada a la economía por el mismo en 1944. En segundo lugar surgió el análisis del input-output, que fue desarrollado por Leontief en el período 1936-1941, [54]. Finalmente aparece la *programación lineal* (PL) desarrollada por George B. Dantzig [47] en 1947.

La PL, denominación introducida por Tucker, fue desarrollada y aplicada por primera vez en 1947 por un grupo de investigadores del Departamento de la Fuerza Aérea de los Estados Unidos, dirigidos por Dantzig. Este grupo fue el encargado de

investigar por mandato de la fuerza armada la posibilidad de aplicar técnicas matemáticas y otras a los problemas de planeación y programación militar. Esta misión de investigación se le encargó al grupo dirigido por Dantzig durante la Segunda Guerra Mundial, cuando el despliegue del personal, de las armas y vehículos por todo el mundo generaban gastos incalculables a la fuerza armada norteamericana. El objetivo era reducir los gastos pero que al mismo tiempo estuviera garantizada la victoria.

Como resultado de sus investigaciones, Dantzig propuso que las interrelaciones entre actividades de una gran organización fueran vistas como un modelo del tipo de PL, y que el programa optimizador se determinara al minimizar una función lineal del objetivo.

El planteamiento matemático general de la PL junto con el método simplex desarrollado por Dantzig constituyó uno de los logros más importantes de la economía lineal de aquella época, porque a partir de los logros obtenidos no se hizo esperar la aplicación de la PL en otros campos, sobre todo en la reconstrucción de las empresas comerciales, industriales y agrícolas devastadas por la guerra.

El modelo general de un problema de PL es

$$\begin{aligned} \max \quad & z = \sum_{j=1}^{m} c_j x_j \\ \text{sujeto a:} \quad & \sum_{j=1}^{m} a_{ij} x_j \le b_i, \quad i = 1,\ldots,m \qquad (1.1a) \\ & x_j \ge 0, \quad j = 1,..,n \end{aligned}$$

donde, si el problema es de una empresa productiva con n líneas de producción y en la que se requiere m tipos diferentes de recursos:

z es el costo (utilidad) total, denominado la *función objetivo*

x_j llamada variable de decisión, es la cantidad (a decidir) de producción del producto j.

c_j es la utilidad unitaria del producto j llamada *costo j*.

b_i es la cantidad del recurso i disponible para la producción, llamada *recurso i*.

a_{ij} es la cantidad del recurso i que se requiere para producir una unidad del producto j, llamada *coeficiente tecnológico ij*.

Este modelo puede escribirse de forma simplicada de la manera siguiente:

$$\begin{aligned} &\max \quad z = cx \\ &\text{Sujeto a:} \quad Ax \leq b \\ &\qquad\quad x \geq 0 \end{aligned} \tag{1.1b}$$

donde

$A=[a_{ij}]$, es la matriz de coeficientes tecnológicos con m filas y n columnas.

$b^T=[b_1,...,b_n]$, vector, n dimensional, de recursos.

$c=[c_1,...,c_m]$, vector, m dimensional, de costos (utilidades).

$x^T=[x_1,...,x_n]$, vector, n dimensional, de las variables.

Uno de los problemas que sirvieron para explicar el procedimiento del método simplex fue el problema de la dieta, planteado por Jerome Cornfield en 1941 (planteado en un memorandum no publicado, pero resuelto por primera vez mediante la programacíon lineal por Stigler, [54]). La importancia de este problema radica en su aplicabilidad práctica dentro de la economía, como por ejemplo en la teoría del consumo. Otros campos de la economía donde se aplica la PL son la teoría del equilibrio [54], el análisis insumo-producto, etc. En general, la PL tiene su aplicación en todas aquellas situaciones que se presentan tanto en empresas individuales como colectivas, nacionales e internacionales, e incluso en otros ámbitos, en las que se desea determinar el mejor plan para alcanzar unos objetivos determinados pero utilizando los recursos disponibles muy limitados.

En forma específica, podemos citar a modo de ejemplo la aplicación de la PL [69] en la industria automovilística, en la industria de productos químicos, en las textilerías, en los sectores de transporte, en flujos de redes (redes de tuberías, redes ferroviarias...), en la teoría de juegos, en la planificación y la elección de proyectos de inversión, etc.

La ampliación de las empresas y la escasa estabilidad de la economía, muy interrelacionada a nivel mundial, así como el modelo actual de economía de mercado, entre otros factores, han hecho que en los últimos 30 años la aplicación de la PL tal como fue desarrollada por Dantzig, ya no sea suficiente o en algunos casos, de imposible aplicación, aun dentro de situación de optimización lineal.

Esto se debe a que la PL desarrollada por Dantzig opera con parámetros conocidos. Sin embargo, en los problemas actuales no se conocen con precisión muchos parámetros,

y por tanto, no se pueden utilizar un modelo y un método diseñados sólo para abarcar parámetros precisos. En el caso de seguir utilizando la PL cuando los parámetros no son precisos tendríamos que presuponer que son exactos (lo cual no expresa la realidad), y los resultados obtenidos de este modo no serían fiables.

En muchos casos, la información que se tiene de los parámetros es imprecisa, por ejemplo en el supuesto de un determinado recurso se puede tener como información que "se podrá adquirir más o menos unas 2000 unidades, pero no más de 2300", o en este otro: "se espera que la venta de un producto determinado nos proporcionará una utilidad aproximadamente de 100 unidades monetarias, o quizás menos, pero no inferior a 70". El utilizar la PL en estos casos, con valores 2000 y 100 o 2300 y 70 o cualquier combinación, podría conducir a un resultado erróneo.

Esta situación y similares limitaciones que se daban en otros campos, vinieron a demostrar la carencia de una teoría matemática para situaciones imprecisas en general. Será Zadeh [216] en 1965, el autor que llene este vacío existente, con su Teoría de Conjuntos Difusos, cuya aplicación en investigación de operaciones, ciencias administrativas, sistemas expertos, inteligencia artificial, teoría de control y en otros campos no se hizo esperar.

El mismo Zadeh junto con Bellman [13] en 1970, desarrolló una primera aplicación de la teoría difusa en la PL en su artículo "*Toma de decisión en ambiente difuso*". Pero fueron Tanaka, Okuda y Asai [190] en 1974, seguido por Zimmermann [222], Negoitia y Sulari [147] en 1976, quienes publicaron los primeros trabajos de aplicación de la teoría difusa en la PL, dando origen así a lo que se ha dado en llamar la *programación lineal difusa* (PLD). Posteriormente se han hecho múltiples estudios sobre este nuevo campo.

Aquí recogemos los resultados más importantes alcanzados hasta la actualidad dentro de la PLD, tanto en los modelos como en los métodos correspondientes, que presentamos en las siguientes secciones. En la sección 1.2 están comprendidos los modelos, en la sección 1.3 los métodos de solución correspondientes y en la sección 1.4 las principales extensiones y aplicaciones.

1.2 MODELOS DE LA PLD

Los modelos de PLD se pueden clasificar con arreglo a muy diversos criterios, por ejemplo, teniendo en cuenta la forma de los modelos, los métodos de solución, o el tipo de parámetros. Al hacer una revisión de la literatura existente se ha elaborado una clasificación de los modelos de PLD, basándonos fundamentalmente en la clasificación que realizan Verdegay [204] y Lai y Hwang [105], en modelos con el conjunto factible difuso, modelos con metas difusas, modelos con coeficientes de la función objetivo difusos, modelos con coeficientes de la matriz tecnológica y recursos difusos, y modelos completamente difusos.

A continuación pasamos a analizar cada uno de estos modelos.

1.2.1 Modelos con el conjunto factible difuso (restricciones difusas)

Desde el punto de vista general, un conjunto difuso puede ser definido de diferentes maneras. Sin embargo, en este contexto el conjunto factible difuso se deriva de considerar los recursos imprecisos, es decir, debido a que los recursos no son conocidos con precisión. En tal situación, para cada recurso, se considera una cantidad deseable b, pero se acepta la posibilidad de que sea mayor hasta un tope máximo $b+t$ (t es denominado nivel de tolerancia). Un modelo que incluye las restricciones con estas característica se representa de la siguiente forma:

$$\begin{aligned} \max \quad & z = cx \\ \text{s. a} \quad & Ax \leq_f b \\ & x \geq 0 \end{aligned} \qquad (1.2)$$

donde el símbolo "$\leq_f$" indica la imprecisión de las restricciones y significa que para cada restricción i, dado el nivel de tolerancia t_i, a cada punto(vector n-dimensional) x se le asocia un número $\mu_i(x) \in [0,1]$ denominado grado de cumplimiento de la restricción i definido por:

$$\mu_i(x) = \begin{cases} 1 & (Ax)_i < b_i \\ f_i((Ax)_i) & b_i \leq (Ax)_i \leq b_i + t_i \\ 0 & (Ax)_i > b_i + t_i \end{cases} \qquad (1.3)$$

donde $f(.) \in [0,1]$, es una función continua y monótona no creciente. Si $\alpha_i = \mu_i(x)$, para $i=1,...,m$, y x fijo, entonces $\alpha = \text{mín}_{i=1,...,m} \{\alpha_i\}$ es el grado de cumplimiento de las restricciones ó el grado de pertenencia de x al conjunto factible difuso.

1.2.2 Modelos con metas difusas

Un problema de PL (caso tradicional) con metas es aquel en el que se fija un valor c_0, llamado meta, que indica, para el caso de maximización, lo que el decisor espera alcanzar como mínimo, y en el caso de minimización, el máximo valor que espera lograr para la función objetivo. Es decir, el conjunto meta es $[c_0,\infty]$ y $[-\infty,c_0]$ para la maximización y minimización respectivamente.

Un modelo de PLD con meta difusa es aquel cuyo conjunto meta es difuso, es decir, que admite que el valor de la función objetivo sea ligeramente inferior a la meta mínima cuando se trata de un problema de maximización, y análogamente para el de minimización. El modelo correspondiente se expresa de la siguiente manera:

$$\begin{aligned} \tilde{\max} \quad & z = cx \\ \text{s. a} \quad & Ax \le b \\ & x \ge 0 \end{aligned} \qquad (1.4)$$

Si t_0 es la cantidad máxima en que la función objetivo debe ser inferior a la meta mínima c_0, entonces a cada vector x se le asocia un número $\mu_0(x)$, que representa el grado en que el decisor considera que se alcanza la meta, definido mediante la función siguiente:

$$\mu_0(x) = \begin{cases} 1 & cx > c_0 \\ f(cx) & c_0 - t_0 \le cx \le c_0 \\ 0 & cx < c_0 - t_0 \end{cases} \qquad (1.5)$$

donde $f(.) \in [0,1]$ es una función continua monótona no decreciente.

1.2.3 Modelos con coeficientes de la función objetivo difusos

Son aquellos cuyos costos o utilidades se conocen vagamente (con imprecisión, no con exactitud). Por tanto se representan mediante números difusos, y el modelo correspondiente, como sigue:

$$\begin{aligned} \max \quad & z = \tilde{c}x \\ \text{s. a} \quad & Ax \leq b \\ & x \geq 0 \end{aligned} \tag{1.6}$$

Evidentemente z también es un número difuso, pero x puede ser un vector de números difusos o no difusos, y cada costo difuso es descrito por su correspondiente función de pertenencia $\mu_j(x)$.

1.2.4 Modelos con coeficientes tecnológicos y recursos difusos

Cuando la información que se tiene de los coeficientes tecnológicos y de los recursos no es exacta se está en el caso de coeficientes y recursos imprecisos, dichos parámetros se consideran como números difusos y entonces, esta información se representa mediante una función de pertenencia para cada parámetro. La representación del modelo asociado a esta situación es de la manera siguiente:

$$\begin{aligned} \max \quad & z = cx \\ \text{s. a.} \quad & \tilde{A}x \leq^{f} \tilde{b} \\ & x \geq 0 \end{aligned} \tag{1.7}$$

donde el símbolo "$\leq^{f}$" es una relación de orden cuya expresión lingüística es "casi menor o igual que" o "esencialmente menor que" entre números difusos, que es una extensión de la relación de orden "$\leq$" definida en los número crips.

1.2.5 Modelos completamente difusos

Estos modelos constituyen el caso general donde todos los parámetros son imprecisos, su representación general es el siguiente:

$$\begin{aligned} \max \quad & z = \tilde{c}x \\ \text{s. a.} \quad & \tilde{A}x \leq^{f} \tilde{b} \\ & x \geq^{f} 0 \end{aligned} \tag{1.8}$$

donde $\leq^{f}$ ($\geq^{f}$) es la relación difusa correspondiente a $\leq$ ($\geq$) como en el modelo (1.7), y los coeficientes difusos están definidos por las siguientes funciones de pertenencia:

$\mu_{c_j}(.)$: función de pertenencia del costo difuso $\tilde{c}_j$.

$\mu_{a_{ij}}(.)$: función de pertenencia del coeficiente difusos $\tilde{a}_{ij}$.

$\mu_{b_i}(.)$: función de pertenencia del recurso difuso $\tilde{b}_i$.

Las funciones de pertenencia de todos los números difusos que aparecen en estos modelos que hemos citado, se especifican en el apartado siguiente (1.3) donde se abordan los correspondientes métodos de solución.

1.3 MÉTODOS DE LA PLD

Existen varios métodos de solución para cada uno de los modelos presentados en la sección anterior. Las diferencias entre ellos se dan o bien en la forma de representación de los parámetros difusos o bien en el procedimiento mismo. Como es natural , hacemos revisión, con u poco de detalle, sólo de los métodos más importantes y comentamos brevemente los otros. Puesto que la información en dichos modelos es imprecisa, también la solución obtenida será imprecisa, obteniéndose en consecuencia, no soluciones en el sentido tradicional, sino soluciones difusas.

1.3.1 Métodos de solución de los modelos con restricciones difusas.

Recogemos dos métodos de solución de modelos con restricciones difusas, el primero propuesto por Tanaka Okuda y Asai [190] y el segundo propuesto por Verdegay [203].

1.3.1.1 Aproximación de Tanaka, Okuda y Asai

Este es el método que se utilizó por primera vez (1974) para resolver el modelo (1.2) correspondiente a la PLD con restricciones difusas.

En esta aproximación, la función f mencionada en (1.3) es lineal, por lo que la función de pertenencia quedaría reflejada de la siguiente manera:

$$\mu_i(x) = \begin{cases} 1 & (Ax)_i \le b_i \\ 1 - \dfrac{((Ax)_i - b_i)}{t_i} & b_i < (Ax)_i \le b_i + t_i \\ 0 & (Ax)_i > b_i + t_i \end{cases} \tag{1.9}$$

y cuya gráfica se muestra en la Figura 1.1

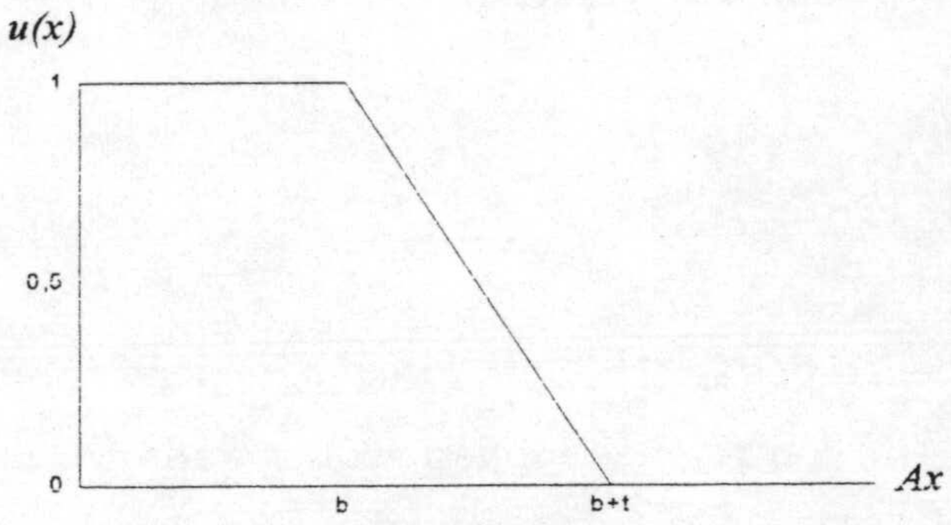


Figura 1.1. Función de pertenencia de las restricciones

Esta aproximación concluye que la solución óptima del modelo (1.2) está dada por la pareja

$$(\alpha^*, x^*) \in (0,1] \times R^n,$$

tal que

$$\alpha^* \wedge f(x^*) = \sup_{\alpha} \{\alpha \wedge \text{máx}_{x \in X_\alpha} f(x)\}$$

donde: $f : R^n \to [0,1]$ es la función objetivo normalizada como sigue :

$$f(x) = \begin{cases} 1 & cx > M \\ \dfrac{cx}{M} & cx \le M \end{cases} \tag{1.10}$$

con M el valor de solución óptima del modelo (1.2) sin considerar la difusidad de las restricciones; y

$$X_\alpha = \{x \in R^n / \wedge_i \mu_i(x) \geq \alpha\},\ \alpha \in [0,1].$$

Asumiendo la unicidad de α^*, y debido a la continuidad de f, (α^*, x^*) se obtiene mediante el siguiente algoritmo:

1) Para $k=1$, $\alpha_k \in (0,1]$.

2) Calcular: $f_k = \max \{ f(x) / x \in X_{\alpha_k} \}$

3) Calcular: $\varepsilon_{k+1} = \alpha_k - f_k$. Si $|\varepsilon_k| > \varepsilon$ ir al paso 4. Si $|\varepsilon_k| < \varepsilon$ ir al paso 5. Con ε alguna precisión deseada.

4) Calcular: $\alpha_{k+1} = \alpha_k - r_k \varepsilon_k$, $r_k \geq 0$ elegido tal que $0 \leq \alpha_{k+1} \leq 1$, ir al paso 2.

5) Hacer $\alpha^* = \alpha_k$ y determinar un óptimo $x^* \in R^n$ tal que $f(x^*) = \max \{f(x) / x \in X_{\alpha^*}\}$.

1.3.1.2 Aproximación de Verdegay

En la aproximación de Verdegay (1982) la función de pertenencia también es lineal, por lo tanto, definida por (1.9) con represenatción gráfica dada por la Figura 1.1..

La interpretación asociada con dicha función de pertenencia la sintetizamos así:

- Cuando $(Ax)_i \leq b_i$, las restricciones se cumplen en forma satisfactoria, por lo tanto el grado de satisfactibilidad es máximo ($\mu_i(x)=1$).
- Cuando $(Ax)_i > b_i + t_i$ se viola la tolerancia, por tanto el grado de satisfactibilidad es nulo ($\mu_i(x)=0$).
- En el caso de $b_i < (Ax)_i \leq b_i + t_i$ se acepta la violación de la restricción, y el grado de satisfactibilidad va disminuyendo a medida que se aleja de b_i.

Lo ideal es hallar la solución óptima para $\mu_i(x)=1$, $i=1,..,m$; sin embargo, sería aceptable obtener una solución para algún valor $\mu_i(x)$ mayor que un α entendida como nivel de satisfactibilidad mínimo fijada a priori de acuerdo con la naturaleza del problema, y lo que es fundamental, en interacción con el decisor.

Por tanto, si $\mu_i(x) \geq \alpha$, *para* $0 \leq \alpha \leq 1$, entonces

$$1 - \frac{(Ax)_i - b_i}{t_i} \geq \alpha \qquad \text{o} \qquad (Ax)_i \leq b_i + (1-\alpha)t_i \quad , \forall i.$$

y si $1-\alpha = \theta$, el problema (1.2) se transforma en el problema paramétrico clásico siguiente:

$$\begin{aligned} \max \quad & z = cx \\ \text{s. a.} \quad & (Ax)_i \leq b_i + \theta t_i, \quad \forall i \\ & x \geq 0, \quad \theta \in [0,1] \end{aligned} \tag{1.11}$$

cuya solución se obtiene mediante métodos clásicos.

Posteriormente, se han desarrollado métodos más generales para resolver problemas del tipo (1.2) con funciones de pertenencia no lineales. Delgado, Herrera, Verdegay y Vila [50] proponen un método análogo al análisis de post-óptimalidad. Ellos demuestran que si $x(\beta)$ es la solución del problema (1.2) con una función de pertenencia lineal f, cuando las funciones de pertenencia g son no lineales continuas la solución de (1.2) es

$$x(r^{-1}(\alpha))$$

donde $r(.)=g(f^{1}(.))$.

1.3.2 Métodos de solución de modelos con funcion objetivo y restricciones difusas

Un caso más general del modelo (1.4) es el siguiente:

$$\begin{aligned} \tilde{\max} \quad & z = cx \\ \text{s. a} \quad & Ax \leq_f b \\ & x \geq 0 \end{aligned} \tag{1.12}$$

La particularización de las restricciones al caso crisp transforma el modelo (1.12) en el modelo (1.4), razón por la cual se han desarrollado métodos para el módelo (1.12) y que pueden ser utilizados en la solución del modelo (1.4). Aquí recogemos tres métodos de

solución, que se diferencian fundamentalmente en la concepción de la imprecisión de la función objetivo.

Puesto que el valor de la función objetivo depende del dominio, conjunto factible en este caso, su característica difusa puede ser debido a la imprecisión de las restricciones o debido a la imprecisión de su meta. En consecuencia pueden darse dos situaciones, una en que no se conoce a priori la meta ni su tolerancia, y otra, cuando se conoce la meta, la tolerancia y la función de pertenencia; para el primer caso tenemos las aproximación de Zimmermann y para el segundo caso tenemos por un lado la aproximación de Chanas y por otro lado la aproximación de Werners.

1.3.2.1 Aproximación de Zimmermann

Para resolver el problema (1.12) según la aproximación de Zimmermann (1976) [222], la meta b_0 y su correspondiente tolerancia t_0 de la función objetivo difusa han de ser dadas inicialmente así como los recursos b_i y sus respectivas tolerancias t_i que se suponen datos del problema.

Teniendo en cuenta estos datos, Zimmermann transforma el problema (1.12) en el siguiente modelo:

$$\begin{aligned} &\text{hallar} \quad x \\ &t\text{al que} \quad cx \geq_f b_0 \\ &\qquad\quad (Ax)_i \leq_f b_i \quad \forall i \\ &\qquad\quad x \geq 0 \end{aligned} \tag{1.13}$$

donde las restricciones difusas están definidas por las funciones de pertenencia lineales que se expresan a continuación:

$$\mu_0(x) = \begin{cases} 1 & si \;\; cx > b_0 \\ 1 - \dfrac{(b_0 - cx)}{t_0} & si \;\; b_0 - t_0 \leq cx \leq b_0 \\ 0 & si \;\; cx < b_0 - t_0 \end{cases} \tag{1.14}$$

$$\mu_i(x)=\begin{cases}1 & si \quad (Ax)_i < b_i \\ 1-\dfrac{((Ax)_i - b_i)}{t_i} & si \quad b_i \le (Ax)_i \le b_i + t_i \\ 0 & si \quad (Ax)_i > b_i + t_i\end{cases} \tag{1.5}$$

y sus correspondientes gráficas se aprecian en las Figuras 1.2 y 1.3

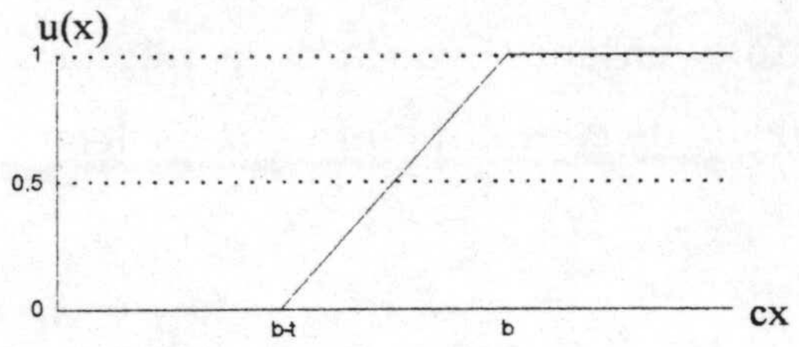


Fig. 1.2: Función de pertenencia de la función objetivo.

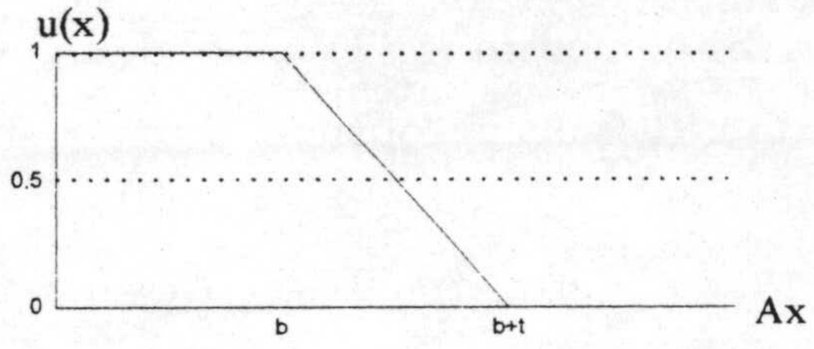


Fig. 1.3: Funciones de pertenencia de las restricciones.

Luego, usando el operador max-min de Bellman y Zadeh, la solución se obtiene a través de la siguiente expresión:

$$\text{máx } \mu_D(x) = \text{máx}\{\text{ mín } [\mu_0(x), \mu_1(x),\ldots, \mu_m(x)]\} \tag{1.16}$$

donde μ_D es la función de pertenencia del espacio de decisión D.

Si $\mu_D(x) = \alpha$, el modelo (1.13) vía (1.16) es equivalente a

$$\begin{aligned} \max \quad & \alpha \\ \text{tal que} \quad & cx \ge b_0 - (1-\alpha)t_0 \\ & (Ax)_i \le b_i + (1-\alpha)t_i, \quad \forall i \\ & x \ge 0 \quad y \quad \alpha \in [0,1] \end{aligned} \tag{1.17}$$

un modelo paramétrico de tipo clásico, que se puede resolver mediantes métodos clasicos.

1.3.2.2 Aproximación de Chanas

Chanas (1983) [35] considera que, debido al poco conocimiento de la región factible difusa, la meta b_0 y la tolerancia t_0 de la función objetivo difusa no podrían ser especificadas inicialmente, por lo tanto se tendrían que determinar con la ayuda del decisor. Para ello, primeramente hay que resolver (1.12), sin considerar la difusidad de la función objetivo, es decir, hay que resolver el modelo equivalente a (1.11).

Si el modelo (1.11) tiene alguna solución factible, entonces para cada número θ, existe $x^*(\theta)$(solución óptima), y una restricción con grado de cumplimiento 1-θ, es decir,

$$\mu_c(x^*(\theta)) = \min_i \{\mu_i(x^*(\theta))\} = 1-\theta\,. \qquad (1.18)$$

Luego, utilizando estas soluciones $z(x^*(\theta))$ y $x^*(\theta)$, el decisor podrá elegir la meta b_0, la tolerancia t_0, y construir la correspondiente función de pertenencia que define la función objetivo difusa dada en el modelo (1.12). Así, se obtiene dicha función de pertenencia, la cual depende de θ, definida de la siguiente manera:

$$\mu_0(x^*(\theta)) = \begin{cases} 1 & si \quad cx^*(\theta) > b_0 \\ 1 - \dfrac{(b_0 - cx^*(\theta))}{t_0} & si \quad b_0 - t_0 \le cx^*(\theta) \le b_0 \\ 0 & si \quad cx^*(\theta) < b_0 - t_0 \end{cases} \qquad (1.19)$$

La solución óptima de (1.12) es $z^*(x^*(\theta))$ y $x^*(\theta)$, donde θ^* es la solución de la ecuación $\mu_0(\theta^*) = \mu_c(\theta^*)$.

1.3.2.3 Aproximacion de Werners

Al igual que Chanas, Werners (1987) [208] considera la difusidad de la función objetivo determinada por la difusidad de las restricciones, pero difiere de aquél en la forma de construir la meta, tolerancia y la función de pertenencia.

Werners define la meta como $b_0 = z^1$, la tolerancia como $t_0 = z^1 - z^0$ donde

$$z^0 = \max\ cx \qquad\qquad z^1 = \max\ cx$$
$$s.a.\quad (Ax)_i \le b_i\ \forall i \qquad\qquad s.a.\quad (Ax)_i \le b_i + t_i\ \forall i$$
$$x \ge 0 \qquad\qquad x \ge 0$$

y la función de pertenencia de la función objetivo como sigue:

$$\mu_0(x) = \begin{cases} 1 & si \quad cx > z^1 \\ 1 - \dfrac{(z^1 - cx)}{z^1 - z^0} & si \quad z^0 \le cx \le z^1 \\ 0 & si \quad cx < z^0 \end{cases} \tag{1.20}$$

En consecuencia, el modelo (1.12) es equivalente al modelo (1.17), como en el caso de la aproximación de Zimmermann.

Otros autores han propuestos modelos que aparentemente pueden parecer diferentes pero, sin embargo, se pueden reducir a este caso y resolverse mediante uno de estos tres métodos explicados aquí. Cabe destacar por ejemplo las propuestas de Zhao, Govind y Fan [219] para resolver problemas con meta difusa y restricciones difusas, algunas de las cuales pueden ser restricciones de igualdad difusa. Estas restricciones de igualdad se transformarán en desigualdades difusas. Los autores citados, en las restricciones de igualdad difusas, consideran las funciones de pertenencias simétricas trapezoidales no lineales. Además, para la toma de decisión difusa usando el operador mínimo, proponen que se considere el mínimo obtenido a partir de las restricciones y el mínimo requerido por el decisor.

1.3.3 Métodos de solución de modelos con coeficientes de la función objetivo difusos

Se dijo, en la formulación del correspondiente modelo (1.6), los coeficientes difusos hacen que la función objetivo también sea difusa. La representación, es decir, la función de pertenencia de la función objetivo depende de la representación de los coeficientes,

que por cierto tiene mulples formas, y de la forma como se relacionan éstas. La diferencia entre los métodos de solución de modelos de la forma (1.6) propuestos radican en estas dos características, que son la forma como se representan los coeficientes difusos y la forma como se relacionan para representar la función objetivo difusa.

Como se ha podido apreciar en las secciones anteriores, los métodos de solución de los modelos de PLD, usando la representación de la difusidad, transforman estos en modelos clásicos. Esta misma idea se mantiene en esta y en las siguientes secciones.

Sin embargo, de aquí en adelante, cuando se trabaja con los coeficientes que son números difusos, se presentan algunas diferencias con los metodos anteriores debido a que los números difusos no se conciben de una manera única. Esta situación conduce a la existencia de métodos generales que engloban todas las concepciones, y otros métodos particulares en las que se utilizan números difusos con caracterísiticas particulares.

Aquí recogemos un enfoque general, el propuesto por Verdegay, y dos enfoques particulares propuestos por Tanaka y otros, y Rommelfanger y otros respectivamente.

1.3.3.1 Aproximación de Verdegay

Verdegay (1984) [202] propone transformar el problema difuso (1.6) en un problema paramétrico clásico, utilizando α-cortes en la función de pertenencia de la función objetivo difusa. En este método, la función de pertenencia de la función objetivo se define como el ínfimo de las funciones de pertenencia de los coeficientes difusos.

Esta misma idea, Delgado, Verdegay y Vila [53], la aplican para el caso en que cada coeficiente está definido por una función de pertenencia de la siguiente manera:

$$\mu_j : R \to [0,1], \qquad \mu_j(x) = \begin{cases} 0 & si \;\; x < a_j \;\; o \;\; x > b_j \\ \underline{h}_j(x) & si \;\; a_j \le x \le c_j \\ \overline{h}_j(x) & si \;\; c_j \le x \le b_j \end{cases}$$

donde: $[a_j,b_j]$ es el soporte del número difuso $\widetilde{c}_j$, $\underline{h}_j$ es continua y estrictamente creciente, $\overline{h}_j$ es continua y estrictamente decreciente y $\underline{h}_j(c_j)=\overline{h}(c_j)=1, \quad \forall j.$

Verdegay [202] muestra la equivalencia del problema (1.6) con el siguiente:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} r_j x_j \\ \text{sujeto a:} \quad & \mu_j(r_j) \geq 1-\alpha, \forall j \\ & Ax \leq b, \\ & x \geq 0, \quad \alpha \in [0,1], \quad r_j \in R \end{aligned} \tag{1.21}$$

Puesto que

$$\mu_j(x) \geq 1-\alpha \quad \Leftrightarrow \quad \underline{h}_j^{-1}(1-\alpha) \leq x \leq \overline{h}_j^{-1}(1-\alpha)$$

el problema (1.21) se transforma en un problema con función objetivo cuyos coeficientes son intervalos para los que, básandose en los resultados de Bitran (1980) [17], Delgado, Verdegay y Vila [53] proponen que la solución se puede obtener resolviendo el problema paramétrico multiobjetivo siguiente:

$$\begin{aligned} \max \quad & \left[c^1 x, c^2 x, \ldots, c^N x\right] \\ \text{sujeto a:} \quad & Ax \leq b, \quad x \geq 0 \\ & c^k = (d_1, d_2, \ldots, d_n), k = 1, \ldots, N, \quad N = 2^n \end{aligned} \tag{1.22}$$

donde cada d_j es $\underline{h}^{-1}(1-\alpha)$ o $\overline{h}^{-1}(1-\alpha), \forall j.$

haciendo uso de un método clásico.

1.3.3.2 Aproximación de Tanaka, Ichihashi y Asai

Esta aproximación considera que los coeficientes de la función objetivo son números difusos triangulares. También la función objetivo se considera un número difuso y con la función de pertenencia definida así:

$$\mu(y)=\begin{cases}1-\left[\!\left[\,2y-(a+b)x\,\right]\!\right]/(b-a)x, & si\ x>0,\ y>0\\ 1 & si\ x=0,\ y>0\\ 0 & si\ x=0,\ y=0\end{cases}$$

donde $b=(b_1,\ b_2,\ldots,\ b_n)$ y $a=(a_1,\ a_2,\ldots,\ a_n)$ tal que $(a_j,\ c_j,\ b_j)$ es la representación triangular del número difuso $\widetilde{c}_j\ \forall j$.

Según Tanaka, Ichihashi y Asai [188], maximizar la función cx es equivalente a maximizar w_1ax+w_2bx, donde $w_1+w_2=1$, $w_1,w_2\in[0,1]$, por tanto la solución del problema (1.6) se obtiene resolviendo el siguiente problema auxiliar:

$$\begin{aligned}&\max && w_1ax+w_2bx\\ &\text{sujeto a:} && Ax\le b,\\ & && x\ge 0,\quad w_1+w_2\in[0,1]\end{aligned}\qquad(1.23)$$

de tipo clásico.

1.3.3.3 Aproximación de Rommelfanger, Hanusheck y Wolf

Estos autores representan los coeficientes difusos de la función objetivo por medio intervalos anidados, determinados por los grados de pertenencia $\alpha_k\in(0,1]$, $k\in M=\{1,\ 2,\ \ldots,p\}$ como sigue:

$$\widetilde{c}_j=\left\{\left[\underline{c}_j^{\alpha_k},\bar{c}_j^{\alpha_k}\right]:\alpha_k\in(0,1],k\in M\right\},\quad \forall\, j=1,2,\ldots,n\qquad(1.24)$$

tal que

$$\forall\alpha_1,\alpha_2\in(0,1],\quad \alpha_1\ge\alpha_2\Rightarrow\left[\underline{c}_j^{\alpha_1},\bar{c}_j^{\alpha_1}\right]\subseteq\left[\underline{c}_j^{\alpha_2},\bar{c}_j^{\alpha_2}\right]\qquad(1.25)$$

Sea $D=\{x\in\mathbf{R}^n\ /\ Ax\le b,\ x\ge 0\}$ y

$$\underline{z}_{max}^{\alpha}=\underline{c}_{\alpha}(\underline{x}_{max}^{*})=\text{máx}\left\{\underline{c}_{\alpha}x\ /\quad x\in D\right\},\qquad \underline{z}_{min}^{\alpha}=\underline{c}_{\alpha}(\underline{x}_{min}^{*})=\text{mín}\left\{\underline{c}_{\alpha}x\ /\quad x\in D\right\},$$

$$\bar{z}_{max}^{\alpha}=\bar{c}_{\alpha}(\bar{x}_{max}^{*})=\text{máx}\left\{\bar{c}_{\alpha}x\ /\quad x\in D\right\},\qquad \bar{z}_{min}^{\alpha}=\bar{c}_{\alpha}(\bar{x}_{min}^{*})=\text{mín}\left\{\bar{c}_{\alpha}x\ /\quad x\in D\right\}.$$

donde: $\underline{c}_\alpha = (\underline{c}_1^\alpha, \underline{c}_2^\alpha, ..., \underline{c}_n^\alpha) \qquad \overline{c}_\alpha = (\overline{c}_1^\alpha, \overline{c}_2^\alpha, ..., \overline{c}_n^\alpha), \quad \alpha \in [0,1]$

Definamos ahora

$$f_1(\underline{c}^\alpha x) = \frac{\underline{c}^\alpha x - \underline{z}^\alpha_{min}}{\underline{z}^\alpha_{max} - \underline{z}^\alpha_{min}} \qquad si \quad \underline{z}^\alpha_{min} \le \underline{c}^\alpha x \le \underline{z}^\alpha_{max}$$

$$f_2(\overline{c}^\alpha x) = \frac{\overline{c}^\alpha x - \overline{z}^\alpha_{min}}{\overline{z}^\alpha_{max} - \overline{z}^\alpha_{min}} \qquad si \quad \overline{z}^\alpha_{min} \le \overline{c}^\alpha x \le \overline{z}^\alpha_{max}$$

Luego, según Rommelfanger, Hanusheck y Wolf (1985) [161], la solución de (1.6) se obtiene resolviendo el problema auxiliar:

$$\begin{aligned} &\max \quad \lambda \\ &\text{sujeto a:} \quad f_1(\underline{c}^\alpha x) \ge \lambda; \quad f_2(\overline{c}^\alpha x) \ge \lambda \\ &\qquad\qquad Ax \le b, \quad x \ge 0, \quad \lambda \ge 0 \end{aligned} \tag{1.26}$$

que es equivalente a este otro problema:

$$\begin{aligned} &\max: \quad \lambda \\ &\text{sujeto a:} \quad (\underline{z}^\alpha_{max} - \underline{z}^\alpha_{min})\lambda - \underline{c}^\alpha x \le -\underline{z}^\alpha_{min} \\ &\qquad\qquad (\overline{z}^\alpha_{max} - \overline{z}^\alpha_{min})\lambda - \overline{c}^\alpha x \le -\overline{z}^\alpha_{min} \\ &\qquad\qquad Ax \le b, \quad x \ge 0, \quad \lambda \ge 0. \end{aligned} \tag{1.27}$$

entonces,

$$f_1(\underline{c}^\alpha x^*) = f_2(\overline{c}^\alpha x^*) = \lambda^*$$

donde (λ^*, x^*) es la solución óptima de (1.27).

1.3.4 Métodos de solución de modelos con coeficientes tecnológicos y recursos difusos

Los números difusos de la matriz de los coeficientes tecnológicos y de los recursos, en el modelo (1.7), se asumen que son del tipo LR , tal como proponen Dubois y Prade [55], mientras que la restricción está dada por una relación de orden entre los números difusos.

Debido a que existen muchos métodos de ordenación de los números difusos, el uso de cada uno en particular puede generar un modelo y un tipo de aproximación. Por otro lado, aún dentro de cada relación de orden, el cumplimiento de las restricciones puede tener o no una determinada tolerancia.

A continuación se presenta un método para el caso en que no hay tolerancia y otro en el que sí existe tolerancia.

1.3.4.1 Aproximación de Tanaka, Ichihashi y Asai

Esta aproximación, de Tanaka y Otros (1984) [188], resuelve el caso en que el cumplimiento de las restricciones no admite tolerancia.

Para definir la relación de orden en el conjunto de los números difusos, hay que especificar a priori un número $\beta \in (0,1]$ denominado grado de optimismo, y entonces, la relación de orden sería la siguiente:

$$\tilde{a} >^{\beta} \tilde{b} \quad \Leftrightarrow \quad (a+\overline{a})_r \geq (b+\overline{b})_r \wedge (a-\underline{a})_r \geq (b-\underline{b})_r \quad \forall r \in [\beta,1] \qquad (1.28)$$

donde $(a+\overline{a})_r$ y $(a-\underline{a})_r$ son estremos superior e inferior respectivamente del r-corte de $\tilde{a}$ así como $(b+\overline{b})_r$ y $(b-\underline{b})_r$ de $\tilde{b}$.

Y luego, Tanaka, Ichihashi y Asai demuestran que la solución de (1.7) se obtiene resolviendo el problema auxiliar convencional siguiente:

$$\max: \quad cx$$

s. a :

$$\left[\left(1-\frac{\beta}{2}\right)(a_i+\underline{a}_i)+\left(\frac{\beta}{2}\right)(a_i-\underline{a}_i)\right]x \leq \left(1-\frac{\beta}{2}\right)(b_i+\underline{b}_i)+\left(\frac{\beta}{2}\right)(b_i-\underline{b}_i) \qquad (1.29)$$

$$\left[\left(1-\frac{\beta}{2}\right)(a_i+\overline{a}_i)+\left(\frac{\beta}{2}\right)(a_i-\overline{a}_i)\right]x \leq \left(1-\frac{\beta}{2}\right)(b_i+\overline{b}_i)+\left(\frac{\beta}{2}\right)(b_i-\overline{b}_i)$$

$$x \geq 0$$

1.3.4.2 Aproximación de Delgado, Verdegay y Vila.

Estos autores [52] (1989) , consideran que en el cumplimiento de las restricciones se admite tolerancia, es decir, resuelven, por analogía al modelo (1.2), el problema siguiente:

$$\begin{aligned} \max \quad & z = cx \\ \text{s. a.} \quad & \tilde{A}x \leq_f^f \tilde{b} \\ & x \geq 0 \end{aligned} \tag{1.30}$$

Si t'_i es la tolerancia en la restricción i, interpretada como en la sección 1.3.1, y con función de pertenencia semejante a (1.9), entonces, según Delgado, Verdegay y Vila, (1.30) es equivalente al problema paramétrico siguiente:

$$\begin{aligned} \max \quad & z = cx \\ \text{s. a.} \quad & \sum_{j=1}^{n} \tilde{a}_{ij} x_j \leq^f \tilde{b}_i + t'_i (1-\alpha) \quad i = 1,...m \\ & x \geq 0, \quad \alpha \in [0,1] \end{aligned} \tag{1.31}$$

el cual puede ser resuelto usando el método anterior.

Como se ha comentado al inicio de esta sección 1.3.4, un problema en el que los parámetros de las restricciones son números difusos puede ser resuelto de múltiples maneras. Se obtendrían tantas aproximaciones diferentes como relaciones de orden usadas en las restricciones. La elección de una determinada relación de orden entre los números difusos tendría que recaer en el decisor, quien tendrá en cuenta la naturaleza del problema. Como ilustración de estos casos tenemos los trabajos de Campos y Verdegay [33] quienes proponen el uso de hasta cinco métodos de ordenación de los números difusos para resolver el problema (1.31).

Otra generalización del problema (1.7) consiste en utilizar un determinado método de expresar los números difusos. Herrera, Kovács y Verdegay [79] presentan un método de aproximación para el supuesto de que los coeficientes del problema (1.7) sean números difusos quasitriangulares en los que la relación de orden denominada (g,p)-valuada es un conjunto difuso en el espacio de los conjuntos difusos generados por la

función g^p ($1 \leq p \leq \infty$). Como caso particular proponen aquel en que la función g es la función generadora de la t-norma de Lukasiewicz.

1.3.5 Métodos de solución de modelos completamente difusos

Se presentan a continuación los métodos de solución del modelo (1.8). Se consideran dos casos fundamentales, el caso uniobjetivo y el caso multiobjetivo con metas también difusas.

1.3.5.1 Aproximación de Tanaka y Asai

Tanaka y Asai (1984) [187] proponen un método de solución para los problemas con metas cuyos coeficientes son números difusos y, a su vez, las metas son también difusas. Esta situación queda mejor expresada de la siguiente manera:

$$\begin{aligned} \max: \quad & (\tilde{c}_1 x, \tilde{c}_2, \ldots, \tilde{c}_l x) \geq^f (\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_l) \\ \text{sujeto a:} \quad & \sum_{j=1}^{n} \tilde{a}_{ij} x_j \leq^f \tilde{b}_i, \quad \forall\, i = 1,2,\ldots,m \\ & x_j \geq 0, \quad \forall\, j = 1,2,\ldots,n \end{aligned} \tag{1.32}$$

la cual, según los autores, es equivalente a:

hallar x

que satisfaga las siguientes restricciones:

$$\begin{aligned} & \tilde{c}_{11} x_1 + \tilde{c}_{12} x_2 + \ldots + \tilde{c}_{1n} x_n \geq^f \tilde{p}_1 \\ & \tilde{c}_{21} x_1 + \tilde{c}_{22} x_2 + \ldots + \tilde{c}_{2n} x_n \geq^f \tilde{p}_2 \\ & \vdots \\ & \tilde{c}_{l1} x_1 + \tilde{c}_{l2} x_2 + \ldots + \tilde{c}_{ln} x_n \geq^f \tilde{p}_l \\ & \tilde{a}_{11} x_1 + \tilde{a}_{12} x_2 + \ldots + \tilde{a}_{1n} x_n \leq^f \tilde{b}_1 \\ & \tilde{a}_{21} x_1 + \tilde{a}_{22} x_2 + \ldots + \tilde{a}_{2n} x_n \leq^f \tilde{b}_2 \\ & \vdots \\ & \tilde{a}_{m1} x_1 + \tilde{a}_{m2} x_2 + \ldots + \tilde{a}_{mn} x_n \leq^f \tilde{b}_m \end{aligned} \tag{1.33}$$

o a las restricciones

$$\begin{aligned}
&\widetilde{Y}_1 = \widetilde{B}_1 x_0 + \widetilde{A}_{11} x_1 + \ldots + \widetilde{A}_{1n} x_n \geq^f 0, \\
&\widetilde{Y}_2 = \widetilde{B}_2 x_0 + \widetilde{A}_{21} x_1 + \ldots + \widetilde{A}_{2n} x_n \geq^f 0 \\
&\vdots \\
&\widetilde{Y}_i = \widetilde{B}_i x_0 + \widetilde{A}_{i1} x_1 + \ldots + \widetilde{A}_{in} x_n \geq^f 0 \qquad (1.34) \\
&\vdots \\
&\widetilde{Y}_M = \widetilde{B}_M x_0 + \widetilde{A}_{M1} x_1 + \ldots + \widetilde{A}_{Mn} x_n \geq^f 0
\end{aligned}$$

donde

$$M = l + m,\ x_0 = 1, \quad x_j \geq 0,\ \widetilde{B}_i = \begin{cases} \widetilde{p}_i, i \leq l \\ -\widetilde{b}_{i-l}, i \geq l \end{cases} \quad \text{y} \quad \widetilde{A}_{ij} = \begin{cases} -\widetilde{c}_{ij}, i \leq l \\ \widetilde{a}_{ij}, i > l \end{cases},\ i = 1,2,\ldots,M,\ j = 1,\ldots,n$$

que escrito en forma vectorial es:

$$\widetilde{Y} = \widetilde{A}x \,\widetilde{\geq}\, 0 \qquad (1.35)$$

donde

$$\widetilde{A} = \begin{pmatrix} \widetilde{A}_1 \\ \vdots \\ \widetilde{A}_M \end{pmatrix} = \begin{pmatrix} \widetilde{B}_1 & \widetilde{A}_{11} \cdots \widetilde{A}_{1n} \\ \vdots & \\ \widetilde{B}_M & \widetilde{A}_{M1} \cdots \widetilde{A}_{Mn} \end{pmatrix}$$

$$\widetilde{A}_i = (\widetilde{B}_i \quad \widetilde{A}_{i1} \cdots \widetilde{A}_{in}) = \{\alpha_i = (\alpha_{i0} \cdots \alpha_{in})^t,\ c_i = (c_{i0} \cdots c_{in})\}$$

La expresión difusa $\widetilde{Y}_i = \widetilde{A}_i x,$ está definida mediante la siguiente función de pertenencia:

$$\mu_{Y_i}(y) = \begin{cases} 1 - \dfrac{|y - x^t \alpha_i|}{c_i{}^t |x|}, & si \quad x \neq 0, \\ 1, & si \quad x = 0,\ y = 0, \\ 0, & si \quad x = 0,\ y \neq 0, \\ 0, & si \quad c_i{}^t |x| \leq |y - x^t \alpha_i| \end{cases} \qquad (1.36)$$

donde $|x| = (|x_0|, \ldots, |x_n|)^t$

Todos los números difusos que Tanaka y Asai consideran, son triangulares simétricos representados por (α,c) con soporte [α-c,α+c], mientras que la llamada condición de no

negatividad difusa, que corresponde al término lingüístico "casi positiva", de los números difusos en (1.34) o (1.35), la definen de la siguiente manera:

$$\tilde{Y}_i \geq^f 0 \quad \Leftrightarrow \quad \mu_{Y_i}(0) \leq 1-h, \quad x^t \alpha_i \geq 0 \qquad (1.37)$$

donde h representa el grado de casi positividad, y el mayor de todos los h de (1.34) es el grado de casi positividad fuerte de (1.35).

Utilizando (1.36) y (1.37) en (1.34) se tiene

$$\mu_{Y_i}(0) = 1 - \frac{\alpha_i{}^t x}{c_i{}^t x} \leq 1-h, \quad \alpha_i{}^t x \geq 0, \quad i = 1,2,\ldots,M \qquad (1.38)$$

que equivale a

$$(\alpha_i - hc_i)x \geq 0, \quad i = 1, 2, \ldots, M \qquad (1.39)$$

luego el problema se reduce a hallar el mayor valor h y x que satisfagan (1.39), es decir, (1.32) es equivalente a:

$$\begin{aligned} &\max \quad h \\ &\text{s. a:} \quad (\alpha_i - hc_i)^t x \geq 0, \quad i = 1, 2, \ldots, M \\ &\qquad\quad x \geq 0, \quad 0 \leq h \leq 1 \end{aligned} \qquad (1.40)$$

Debido a que las restricciones son no lineales (por el producto h y x), Tanaka y Asai proponen el siguiente algoritmo para resolverlas:

Algoritmo:

1) Determinar un valor inicial pequeño h tal que exista un conjunto factible que satisfaga (1.40)
2) Sea $\lambda>0$ un pequeño incremento. Encontrar el más pequeño valor $h+(k+1)\lambda$ no compatible con (1.40), $k=0,1,2,....$
3) Para el valor $h+k\lambda$, en el sistema compatible (1.40), seleccionar la desigualdad más interesante. Si eligió la desigualdad j, considere $J=\alpha_{j1}x_1+...+\alpha_{jn}x_n$, como función objetivo auxiliar; luego, resolver:

$$\begin{aligned} &\max \quad J \\ &\text{s. a}: (\alpha_i - (h + k\lambda)c_i)x \geq 0, \quad i = 1,2,...,M \\ &\qquad x \geq 0 \end{aligned} \tag{1.41}$$

La solución $\boldsymbol{x}^*$ del problema (1.41) es la solución aproximada de (1.40) y consecuentemente de (1.32).

1.3.5.2 Aproximación de Carlsson y Korhonen

El punto de partida de Carlsson y Korhonen [30] (1986) es que la solución óptima $z^*=z^*(c,\text{-A},b)$ del problema de PL

$$\begin{aligned} &\max \; z = cx \\ &\text{s. a: } Ax \leq b, \; x \geq 0 \end{aligned}$$

es una función creciente respecto a los parámetros c, -A y b.

Este resultado les permite considerar el problema (1.8), de tal manera que sus parámetros difusos están definidos mediante funciones de pertenencia monótonamente decrecientes.

En general, estas funciones de pertenencia pueden ser lineales o no lineales. Carlsson y Korhonen prefieren una función de pertenencia exponencial, debido a que las funciones exponenciales son muy flexibles, así si p representa a cualesquiera de los parámetros difusos, su función de pertenencia es:

$$\mu_p(t) = \begin{cases} 1 & t \leq p^0 \\ a_p\left[1 - \exp\left[-b_p\left(\dfrac{t - p^1}{p^0 - p^1}\right)\right]\right] & p^0 \leq t \leq p^1 \\ 0 & t \geq p^1 \end{cases} \tag{1.42}$$

donde

$$a_p = \frac{1}{1 - \exp(-b_p)} \qquad \text{y} \qquad b_p \neq 0$$

los números p^0, p^1 y b_p son dados por el usuario(decisor, etc).

En esta definición de la función de pertenencia los autores consideran que en el caso de $\mu_p=1$ ($t \leq p^0$) se tiene un problema no difuso, mientras que si $\mu_p=0$ ($t \geq p^1$), el problema no es aplicable. Por lo tanto, tiene importancia desde el punto de vista de la difusidad, el intervalo $[p^0,p^1]$, y por eso, será para este caso cuando se utilice el presente método.

Si μ_c, μ_A y μ_b representan las funciones de pertenencia de los costos, coeficientes tecnológicos y recursos difusos respectivamente, la solución existirá cuando

$$\lambda = \mu_c = \mu_A = \mu_b \tag{1.43}$$

de donde podemos obtener

$$c=\mu_c^{-1}(\lambda), \qquad A=\mu_A^{-1}(\lambda) \qquad y \qquad b=\mu_b^{-1}(\lambda)$$

donde a su vez $\lambda \in [0,1]$.

Luego el modelo (1.8) puede ser reescrito como sigue

$$\begin{aligned} &\max && [\mu_c^{-1}(\lambda)]x \\ &\text{s. a:} && [\mu_A^{-1}(\lambda)]x \leq \mu_b^{-1}(\lambda) \\ & && x \geq 0, \ \lambda \in [0,1] \end{aligned} \tag{1.44}$$

Este modelo es un modelo paramétrico no lineal que puede resolverse usando para parámetros fijos λ (que en este caso se transforma en un modelo lineal simple). Así se genera un conjunto de soluciones con distintos valores del parámetro que constituyen un conjunto de alternativas de solución para el decisor.

Al igual que para los otros modelos estudiados anteriormente, para los modelos completamente difusos existen otros métodos que no son del todo diferentes a estos dos métodos que acabamos de resumir. En la aproximación de Tanaka y Asai se utilizan los números difusos triangulares, mientras que en el de Carlsson y Korhonen se abarcan

casos generales. Ramik y Rommelfanger [157] presentan un método que, comparado con el método propuesto por Tanaka y Asai viene a ser una extensión del mismo, y comparado con el de Carlsson y Korhonen, constituiría una particularización del método propuesto por estos autores. Ramik y Rommenlfanger proponen un método de solución para el caso en el que los parámetros sean números difusos trapezoidales; desarrollando este mismo método con mayor detalle para los números difusos trapezoidales rectos, e introducen dos nuevos métodos de comparación de números difusos trapezoidales para interpretar las restricciones difusas. Trabajos similares han realizado Nakamura y Gen [143] quienes definen otras relaciones de igualdad y desigualdad de números difusos trapezoidales y luego las utilizan para resolver problemas de PL completamente difusos cuyos parámetros son números difusos trapezoidales.

1.4 EXTENSIONES Y APLICACIONES DE PLD

Como se ha comentado, la PLD nace precisamente allí donde la PL no tiene la suficiente capacidad de resolver los problemas reales, por tanto, ensancha el horizonte de las aplicaciones de la PL en general, ya sea en la solución de problemas reales muy específicos, o como una ampliación de los métodos de los diferentes casos especiales como es la programación multiobjetivos, problemas de transporte, programación entera, teoría de la dualidad o programación estocástica, por citar sólo algunos ejemplos.

Desde que en 1974 Tanaka, Okuda y Asai [190] empezaron a introducir la teoría difusa en la programación matemática, la segunda mitad de la década 70 ha sido el período inicial de construcción de la PLD. Algunos trabajos fundamentalmente resolvían problemas aplicativos como los de Negoita y Sularia [147] (1976), Sularia [186] (1977). Otros como Zimmerman [223] (1978) construyeron métodos generales para la PLD.

Los años 80 constituyeron sin duda alguna la *década de oro* de la PLD, esto naturalmente desde el punto de vista de la construcción de diversos modelos y métodos de solución, las mismas que hemos presentado y comentado en las secciones 1.2 y 1.3. En esta década, también se continuaron con el estudio de aplicaciones y las extensiones.

La presente década, relacionado con la programacíon lineal difusa, está caracterizada por el perfeccionamiento de las extensiones y aplicaciones iniciadas en la década pasada, así como por la introducción en otros nuevos campos. Sugeridos por este hecho, la presentación de la revisión de las extensiones y aplicaciones lo haremos en dos grupos: por un lado, en las secciones 1.4.1, 1.4.2 y 1.4.3 aquéllas que se iniciaron en la década pasada y que fueron consolidados en la primera mitad de esta década y por otro lado, en la sección 1.4.4, las extensiones y aplicaciones iniciadas en esta década y que se están consolidando en esta segunda mitad y de cara al siglo XXI.

1.4.1 Extensiones multiobjetivo

En los problemas de planificación de la producción, no sólo hay un único criterio de optimización, sino que existen casos en los que es necesario optimizar varias funciones objetivo. Cuando tanto las restricciones como las funciones objetivo son lineales nos encontramos ante un problema de PL multiobjetivo (PLMO), que tiene la siguiente forma general:

$$\begin{aligned} \max \quad & Z(x) \\ \text{s. a.} \quad & Ax \le b, \\ & x \ge 0. \end{aligned} \tag{1.45}$$

donde $Z(x) = (z_1(x), z_2(x), \ldots, z_K(x))$ y cada $z_k(x)$ es lineal, $k=1,\ldots,K$.

Cuando no existían métodos específicos de solución para este tipo de problemas, se intentaron resolver mediante los métodos de la PL y concretamente el método simplex. Para ello, basándose en algún criterio, se elegía una de las funciones objetivos y las demás se utilizaban como restricciones adicionales, fijando previamente una meta que hiciera las veces de recursos, con las consiguientes pérdidas de autenticidad. Por ello, se desarrollaron métodos muy específicos para estos casos, entre los que destacan tres: métodos de maximización vectorial [42], programación de metas [42], y técnicas iterativas [226]. Estos métodos, sin embargo, también presentan algunos inconvenientes a pesar de sus ventajas.

Para resolver esos inconvenientes de los métodos convencionales ya citados, a la vista de los logros significativos de la introducción de la teoría difusa en la PL, se intentó introducir la teoría difusa también en la programación multiobjetivo. El primer paso consistió en construir métodos para resolver los problemas PLMO en el sentido clásico haciendo uso de la teoría difusa. El primero de estos métodos fue desarrollado por Zeleny [218] quien transforma en una sola todas las funciones objetivo mediante la combinación convexa, en la que los coeficientes convexos son números difusos cuyas funciones de pertenencia son definidas por el decisor. Otro de los métodos importantes en esta dirección es el método dado por Hannan [78].

Hannan parte de la consideración de que el decisor tiene alguna meta (aunque en forma vaga) respecto a las funciones objetivo, por tanto la meta para cada función objetivo es un conjunto difuso, cuya función de pertenencia discreta la define el propio decisor. A partir de ello construye la función de pertenencia continua y lineal a trozos. Luego presenta tres métodos para construir una sola función objetivo, todos ellos basados en el nivel de aceptación de la meta expresado por el grado de pertenencia, lo cual permite obtener un problema de PL con una sola función objetivo.

El primero en introducir los métodos de la PLD en la PLMO fue Zimmermann [223], definiendo un problema de PLD multiobjetivo (PLDMO), de la siguiente forma

$$\begin{aligned} \tilde{\min} \quad & z_k(x), \quad k = 1,2,\ldots,K \\ \text{s. a.} \quad & Ax \le b, \\ & x \ge 0. \end{aligned} \tag{1.46}$$

Se observa que este problema es del tipo (1.4) dado en la sección 1.2, cuyos métodos de solución se describe en la sección 1.3.2, con la diferencia de que en este caso existen varias funciones objetivo.

Si cada función objetivo difusa (en el sentido de que tiene una meta difusa) está definida por medio de la función de pertenencia lineal $\mu_i(z_i(x))$, según Zimmermann [223], utilizando la decisión difusa de Bellman y Zadeh, el problema (1.46) es equivalente a:

$$\begin{aligned} \max \ & \min_{k=1,\ldots,K} \mu_k(z_k(x)), \\ \text{s. a.} \quad & Ax \le b, \\ & x \ge 0. \end{aligned} \tag{1.47}$$

que puede transformarse en un problema de PL clásica

$$\begin{aligned} &\max \quad \lambda \\ &\text{s. a.} \quad \lambda \le \mu_k(z_k(x)), \quad k=1,\cdots,K \\ &\qquad Ax \le b, \\ &\qquad x \ge 0, \lambda \in [0,1] \end{aligned} \tag{1.48}$$

Se observa que en (1.47) se utiliza el operador mín. Zimmermann también propone el uso del operador producto y así, el correspondiente problema equivalente a (1.46) es:

$$\begin{aligned} &\max \prod_{k=1}^{K} \mu_k(z_k(x)), \\ &\text{s. a.} \quad Ax \le b, \\ &\qquad x \ge 0. \end{aligned} \tag{1.49}$$

este problema (1.49) tiene la función objetivo no lineal y por tanto no se puede resolver mediante los métodos de la PL.

Posteriores trabajos son similares a los métodos propuestos por Zimmermann, pero con algunas alteraciones que permiten mejorar los resultados. Así tenemos los trabajos de Benson [15] , Delgado [49], Fedrizzi, Kacprzyk y Roubens [63], Feng Ying-Jun [65], Ignizio [88] y Narasimhan [144, 145]. Los trabajos de Hannan [78] y Inuiguchi, Ichihashi y Kume [91] utilizando funciones de pertenencia continuas y lineales a trozos, Yang e Ignizio [212] y recientemente Li y Yu [119] utilizando funciones de pertenencia no lineales aproximados por lineales a trozos, Leberling [114] utilizando ciertas funciones de pertenencia no lineales al igual que Buckley [23], con la diferencia de que este último las utiliza para un problema de programación cóncavo o convexo multiobjetivo. Luhandjula [124] sustituye los operadores *min* y producto por el *operador-γ* y, para efectos de la simplificación de cálculos, propone el uso del operador *suma min-acotado*. Chanas [36] a diferencia de Zimmermann considera que no es necesario que el decisor proporcione la función de pertenencia (meta difusa) de cada función objetivo, sino que ésta se construye tomando como referencia el mínimo y el máximo valor que toma cada una de las funciones objetivo en la región de factibilidad. Luego, las funciones objetivo se ordenan y se resuelve después de transformar en un problema paramétrico. Mohamed [138] utiliza el concepto de variables desviacionales para transformar el problema (1.46) al modelo clásico.

Existen casos en que las funciones objetivo no son precisamente lineales sino cociente de funciones lineales. Estos casos son considerados también dentro de la PL y son denominados problemas de programación multiobjetivo fraccionales (PMOF). Luhandjula en [126] introduce por primera vez el uso de la PLD en PMOF, considerando que las metas están dadas mediante variables linguísticas. Este método posteriomente es mejorado por Dutta, Tiwari y Rao [56,58]. También Ohta y Yamaguchi [149] utilizaron la PLD para resolver problemas de PMOF.

Las funciones objetivo de los problemas vistos hasta ahora son todas relativamente independientes, sin embargo en los problemas reales esto casi no sucede porque, de una u otra forma, existen interdependencias entre ellos. En los casos más simples pueden tener sólo diferentes grados de importancia, y en otros casos muy complejos pueden existir objetivos en conflicto. Para el caso de las funciones objetivo con diferentes grados de importancia, es decir, cuando existen prioridades entre los objetivos, se han desarrollado algunos métodos convencionales para resolverlos, como es el caso de Ignicio [89]. Tiwari, Dharmar y Rao introducen la difusidad en los problemas de este tipo: en un primer trabajo [192], utilizan una metodología similar a la de Ignizio, resolviendo subproblemas difusos en las que las soluciones de los subproblemas con funciones objetivo más prioritarios constituyen restricciones, y en [193] utilizan el modelo aditivo con pesos. Cabe anotar también el trabajo de Rubin y Narasimhan [163] dentro de este contexto. La propuesta de Chen [43] es una variación de las anteriores. Lee y Li [115] estudian la PLDMO en que algunas de las funciones objetivo son de minimización y las demás de máximización (el problema con estas características se denomina programación de compromiso); introduciendo la solución *ideal* y la *anti-ideal*, obtienen la solución de *compromiso* minimizando la distancia entre la solución ideal y la solución deseada. Para hallar esta solución de compromiso, basándose en los resultados de Zimmermann [223], Lee y Li proponen una aproximación en dos fases. En la primera usan el operador mínimo y en la segunda, un operador compensatorio como es el promedio aritmético. La aproximación de dos fases propuesta por Lee y Li es uno de los métodos más simplificados para resolver la PLDMO, por ello otros autores han intentado sacar el mejor provecho posible de ella, así Mohamed [138] relaja el método en el sentido de que los pesos no necesariamente tienen que ser iguales, pero deben ser positivos para generar una solución general eficiente. Por otro lado, Ida y Gen [87] mejoran el método

propuesto por Lee y Li proponiendo un algoritmo basado en la unificación de las dos fases.

Otro tipo de problemas de PLDMO son aquellos en los que los parámetros son difusos. Aunque estos problemas ya habían sido tratados por Orlovski [153], Lee y Li [115] utilizando los α-cortes, proponen un método de solución diferente. Dentro de este contexto se incluyen las aportaciones de Wang y Wang [206] que extienden, los métodos desarrollados por Tong [196] para resolver la PL multiobjetivo con parámetros que son intervalos, a la PLDMO con costos difusos.

Una generalización de los problemas multiobjetivo son los problemas de programmación lineal multicriterios y de niveles multirestrictivos (MC^2) introducidos y resueltos por Seiford y Yu [177]. El método para resolverlos se denomina MC^2-simplex, utilizado por Lee, Shi y Yu [116]. Posteriormente, Shi y Liu introducen la teoría difusa, primeramente [179] usando una función de pertenencia para el conjunto potencial de soluciones, y luego [123] utilizando los métodos de la PLD para determinar una solución potencial. Recientemente Cadenas y Verdegay [28] han introducido la posibilidad de que cada objetivo provenga de un decisor diferente.

1.4.2 Extensiones uniobjetivo

En esta sección recogemos las extensiones de los conceptos de la PLD en los problemas especiales como son el problema de transporte, el análisis de sensibilidad, programación entera y otros.

1.4.2.1 Problemas de transporte difusos

Motivado por los diferentes artículos que aparecieron sobre la PLD, Óhéigeartaigh [148] inicia la extensión y aplicación de los métodos de la PLD a problemas de transporte con restricciones difusas. Sobre este trabajo inicial, Chanas, Kolodziejczyk y Machaj [38] hacen las mejoras correspondientes y posteriormente Verdegay [201], Delgado, Verdegay

y Vila [51] y Chanas, Delgado, Verdegay y Vila [37] generalizan los resultados anteriores.

Consideremos el siguiente problema de transporte difuso

$$\begin{aligned} \widetilde{\min} \quad & c(\{x_{ij}\}) = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\ \text{s.a.} \quad & \sum_{j=1}^{n} x_{ij} =_f \widetilde{a}_i, \quad i = 1,\ldots,m \\ & \sum_{i=1}^{m} x_{ij} =_f \widetilde{b}_j, \quad j = 1,\ldots,n \\ & x_{ij} \geq 0, \quad i = 1,\ldots,m; \quad j = 1,\ldots,n \end{aligned} \tag{1.50}$$

donde:

$$\widetilde{a}_i = (a_i^1, \underline{a}_i^1, a_i^2, \overline{a}_i^2), \quad \widetilde{b}_j = (b_j^1, \underline{b}_j^1, b_j^2, \overline{b}_j^2)$$

son números difusos trapezoidales no negativos.

Teniendo en cuenta que las restricciones son difusas, es lógico que la función objetivo (el costo total admisible) también sea difusa. Supongamos que está definida por la siguiente función de pertenencia:

$$\mu_{\widetilde{G}}(x) = \begin{cases} 1 & \text{para } x < c_0 \\ f(x) & \text{para } x \geq c_0 \end{cases}$$

donde $f(x)$ es una función continua, decreciente y $f(c_0)=1$. Un caso particular de $f(x)$ es una función lineal.

Chanas, Kolodziejczyk y Machay [38] proponen hacer uso de la programación paramétrica en forma similar que en la PLD, tal como se detalla a continuación.

Haciendo un (1-α)-corte en los números difusos de (1.50), para cada valor $\alpha \in [0,1]$, se obtiene el siguiente problema:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

$$\text{s.a.} \quad \sum_{j=1}^{n} x_{ij} \in \left[a_i^1 - \alpha \underline{a}_i^1, a_i^2 + \alpha \overline{a}_i^2\right], i = 1,\ldots,m \qquad (1.51)$$

$$\sum_{i=1}^{m} x_{ij} \in \left[b_j^1 - \alpha b_j^1, b_j^2 + \alpha \overline{b}_j^2\right], j = 1,\ldots,n$$

$$x_{ij} \geq 0, \quad i = 1,\ldots,m; \quad j = 1,\ldots,n; \quad \alpha \in [1-\bar{r},1],$$

$$\bar{r} \in \sup_x \mu_{\tilde{a}\cap\tilde{b}}(x), \quad \tilde{a} = \sum_i \tilde{a}_i, \quad \tilde{b} = \sum_i \tilde{b}_i.$$

Se observa que tanto las ofertas como las demandas (restricciones) no son iguales a un número sino que están comprendidas entre dos cantidades (pertenecen a un intervalo de números). El problema (1.51) se puede reducir a un problema clásico de un solo estado. Para ello se consideran los valores del extremo izquierdo de cada intervalo como la demanda mínima o la oferta mínima respectivamente, que necesariamente debe ser satisfecha en el caso de demanda, y necesariamente distribuida en el caso de oferta. Luego la diferencia entre el extremo derecho y el izquierdo de cada intervalo, se debe considerar como la demanda adicional, u oferta adicional, que no necesariamente debe ser satisfecha o distribuida respectivamente. Esto supone la introducción de costos muy elevados, o costos nulos, según si se quiere evitar o no los envíos desde determinados orígenes hacia determinados destinos. Así, queda planteado un problema de transporte paramétrico clásico, cuya solución es una función en términos del parámetro α.

La solución, de acuerdo a la decisión en un ambiente difuso dada por Bellman y Zadeh, se obtiene mediante:

$$\max \quad \mu_{\tilde{D}}(\{x_{ij}(\alpha)\}) = u_{\tilde{G}}(\{x_{ij}(\alpha)\}) \wedge (1-\alpha) \qquad (1.52)$$

El operador $\wedge$ usado en (1.52) puede ser sustituido por cualquier otro operador que sea no decreciente con respecto a cada uno de sus componentes.

Trabajos posteriores son extensiones de los anteriores, como son los trabajos de Bit, Biswal y Alam. En un primer trabajo [18], estos autores resuelven problemas de transporte multiobjetivos (multicriterios) utilizando las técnicas de PLD mientras que en un segundo trabajo [23], resuelven el problema de transporte sólido. Esta denominación se da a aquellos problemas de transporte de objetos de distintos tipos, que implican el uso de diferentes medios de transporte y que incluyen también múltiples objetivos.En un

trabajo publicado recientemente, Jimenez y Verdegay [96] abordan el problema del transporte sólido usando la PLD. Otro trabajo es el de Chalam [34], quien estudia un problema de transporte con demanda aleatoria y cuyas metas son menores de las soluciones mínimas.

En los trabajos previos de problemas de transporte difusos [37,38,51 y 201] no se especifica la condición de ser números enteros para las variables, pese a que desde el punto de vista de las aplicaciones, esta condición es importante. Por otro lado, los métodos propuestos en ellos transforman el problema de transporte difuso en un problema paramétrico clásico.

Recientemente, Chanas y Kuchta [39] han propuesto un método similar a los anteriores para resolver los problemas de transporte difusos en donde las variables tienen la condición de ser números enteros. En este método, el problema difuso se transforma primeramente en un problema paramétrico intervalar de tipo clásico, luego mediante un algoritmo para diversos valores del parámetro se resuelve el problema en forma similar que en [38] transformando en un problema simple de tipo clásico.

Finalmente cabe destacar también los trabajos de Verma, Biswal y Biswas [205], quienes utilizan las técnicas de PL para resolver problemas de transporte multiobjetivo introduciendo metas difusas que definen mediante funciones de pertenencia no lineales, en concreto, mediante funciones hiperbólicas y exponenciales.

1.4.2.2 Dualidad y análisis de sensibilidad

La dualidad y el analisis de sensibilidad, dentro de la PL, están intimamente relacionadas, relación que se extiende también en la PLD al igual que otros conceptos.

Los primeros en extender los conceptos de la *dualidad* en la PLD fueron Hamacher, Leberling y Zimmermann [77], quienes estudiaron los problemas de la PLD con restricciones de dos tipos, algunas difusas y las otras no difusas. Después de transformarlos al problema clásico equivalente, obtuvieron el problema dual no difuso para luego analizar y dar la interpretación económica de las variables duales. Le

siguieron Rodder y Zimmermann [160], introduciendo nuevas interpretaciones económicas de la teoría de la dualidad usando conjuntos difusos.

En la PL clásica es bien conocido que la relación primal-dual está dada de una manera única, pero esto no ocurrre en la PLD, por lo menos dentro del enfoque actual, pues depende del tipo del problema clásico en que se transforme el problema difuso.

Una de las definiciones del problema dual en PLD y su correspondiente método para obtenerlo es la que establece Verdegay [202] basándose en la simetría entre restricciones difusas y una función objetivo difusa tal como fueron definidas por Bellman y Zadeh [13]. Verdegay considera los dos tipos siguientes:

a) Cuando la función objetivo es completamente conocida pero las restricciones son difusas, llámese problema P1.
b) Cuando las restricciones son exactamente conocidas pero los coeficientes de la función objetivo son difusos, llámese problema P2.

Para estos dos tipos de problemas de PLD, P1 y P2 (primal), existen sendos problemas duales, de tal modo que cada primal y su correspondiente dual tienen la misma solución.

Considérese el problema:

$$\max \{cx \,/\, Ax \lesssim_f b,\ x \geq 0\} \qquad (1.53)$$

igual al modelo (1.2). Si μ_i (i=1,...,m), la función de pertenencia de la i-ésima restricción, es definida como

$$\mu_i(t) = \begin{cases} 1 & si \quad t < b_i \\ [(b_i + d_i) - t]/d_i & si \quad b_i \leq t \leq b_i + d_i \\ 0 & si \quad t > b_i + d_i \end{cases} \qquad (1.54)$$

Entonces, Verdegay [202] demuestra que su correspondiente problema dual difuso es:

$$\min \{eu \,/\, uA^T \geq c,\ u \geq 0\} \qquad (1.55)$$

con los coeficientes e (costos) difusos dados por (1.54). Y recíprocamente, si (1.55) es un problema difuso (igual al modelo 1.6) su correspondiente problema dual es (1.53).

Esta construcción del dual se puede generalizar, extendiéndose al caso de la función de pertenencia no lineal estrictamente monótona de los parámetros difusos, y también al caso en que tanto la función objetivo como las restricciones sean difusas, y la dualidad obtenida en todos ellos se conoce como "auto-dualidad".

El *análisis de sensibilidad* es un área muy bien explorada en la PL clásica. Consiste en determinar la solución de un nuevo problema, que representa una pequeña variación con respecto a un problema ya resuelto, utilizando la solución dada para éste. Se puede decir que anteriormente, el análisis de sensibilidad venía a ser una manera rudimentaria de comprender los problemas en los que no se conocían con exactitud los datos, constituyendo una especie de transición hacia la PLD, pero evidentemente no es lo mismo, aunque en realidad, la idea del análisis de sensibilidad en la teoría clásica puede también extenderse a la PLD, como efectivamente ha ocurrido así.

Los primeros que introdujeron el análisis de sensibilidad en la PLD fueron Hamacher, Leberling y Zimmermann [77], quienes analizaron el siguiente problema:

$$\begin{aligned} &\tilde{\max} \quad cx \\ &\text{s. a.} \quad a_i^T x \leq_f b_i, \quad i = 1,\ldots,m_1 \\ &\qquad d_j^T x \leq b'_j, \; j = m_1+1,\ldots,m_1+m_2, \\ &\qquad x \geq 0, \quad m_1 + m_2 = m. \end{aligned} \qquad (1.56)$$

donde c, a_i, d_j son vectores con n componentes y b_i y b'_j vectores con m_1 y m_2 componentes respectivamente.

Utilizando la aproximación de Zimmermann dada en la subsección 1.3.2.1, se obtiene de (1.56) el siguiente problema clásico:

$$
\begin{aligned}
&\max \quad \lambda \\
&\text{s. a.} \quad \lambda p + t \le p \qquad (I) \\
&\qquad Ax - t \le b, \qquad (II) \\
&\qquad\quad t \le p \qquad (III) \\
&\qquad Dx \quad \le b' \qquad (IV) \\
&\lambda \in R,\ x, t \ge 0.
\end{aligned}
\tag{1.57}
$$

donde $A=(a_i^T)$, $b=(b_i)$, $t=(t_i)$, $p=(p_i)$, i=0,1,...,m_1 y $D=(d'_j)$, $b'=(b'_j)$, j=m_1+1,...,m_1+m_2; p_i es la tolerancia de la i-ésima restricción (meta si i=0).

El análisis de sensibilidad que hacen Hamacher, Leberling y Zimmermann es teniendo en cuenta las variaciones de los niveles de tolerancia, es decir, considerando las variaciones en el vector p, por tanto involucran solamente las restricciones (I) y (III) de (1.57). Estos autores realizan dicho análisis considerando tres casos: $\lambda_{max}=1$, $0<\lambda_{max}<1$ y $\lambda_{max}=0$. En el primer caso para $\lambda_{max}=1$, analizan cuando p varía a $p+\Delta p>0$ y demuestran que la solución óptima no sufre ninguna alteración; para los casos segundo y tercero y si λ es una variable básica, obtienen una fórmula en términos del incremento para la variación de la solución, pero no presentan ninguna conclusión para el caso de que λ no sea una variable básica.

Posteriormente Tanaka, Ichihashi y Asai [189] hacen uso del análisis de sensibilidad para determinar una información de un problema de PLD. Ostermark [154] hace el análisis de sensibilidad para establecer la interdependencia entre los parámetros de un PLD, y Fuller [66] demuestra la estabilidad (con la norma C) de la solución de un problema de PLD con respecto a pequeñas variaciones en las funciones de pertenencia de los coeficientes difusos. Ambos autores tratan los problemas de PLD completamente difusos cuyos parámetros tienen representación difusa triangular simétrica.

Dutta, Rao y Tiwari [57] hacen el análisis de sensibilidad de PLD fraccionaria. Después de expresarla en dos problemas del tipo PLD, aplican el análisis realizado por Hamacher, Leberling y Zimmerman.

Por otro lado, García-Aguado y Verdegay [67] proporcionan el rango de variación de la solución óptima de un problema de PLD cuando los niveles de tolerancia del cumplimiento de las restricciones varían dentro de un intervalo.

Finalmente remarcamos la importancia de los resultados de Delgado, Herrera, Verdegay y Vila [50], quienes proponen que, utilizando la solución de un problema de PLD con restricciones difusas donde las funciones de pertenencia son lineales, pueden resolverse el mismo problema pero cuando las funciones de pertenencia son no lineales.

1.4.2.3 Programación entera difusa

Desafortunadamente, existen escasos trabajos que extienden los conceptos de la PLD en la programación entera, más aún, esta situación se acentúa en el caso de la programación entera en general.

Para el caso general, recogemos aquí, la importante aportación de Fabian y Stoica [62], que plantea el siguiente problema:

$$\begin{aligned} &\max \quad cx \\ &s.a. \quad (Ax)_i \le_f b_i, \quad i = 1,\ldots,m \\ &\qquad x \ge 0, \quad \text{y} \quad x \text{ números enteros} \end{aligned} \tag{1.58}$$

si p_i es la tolerancia para la restricción i, y si se asume que las funciones de pertenencia de las restricciones difusas son lineales, para resolver (1.58), Fabian y Stoica proponen resolver el modelo auxiliar convencional siguiente:

$$\begin{aligned} &\max \quad cx + \textstyle\sum_i r_i [b_i - (Ax)_i] w_i \\ &\text{s. a.} \quad (Ax)_i - p_i w_i \le b_i, \quad w_i \in \{0,1\}, \; i = 1,\ldots,m \\ &\qquad x \ge 0, \quad y \quad x \text{ números enteros} \end{aligned} \tag{1.59}$$

donde r_i, i=1,...,m son coeficientes de penalización.

Es evidente que (1.59) es un problema de programación no lineal y en consecuencia, su resolución exige otras técnicas que no son objeto de estudio en este trabajo.

La introducción de los conceptos de la PLD en la *programación entera binaria* ha tenido mejor suerte que el caso general. Desde que iniciaran en ella Zimmerman y Pollatschek [225], les siguieron Herrera y Verdegay [80], Castro, Herrera y Verdegay [31]. El resumen de estos trabajos propuestos podemos encontrar en [82,107].

La aproximación propuesta por Zimmerman y Pollatschek sirva para resolver problemas de programación binaria difusa (PBD) que se pueden expresar en la forma siguiente:

$$\begin{aligned} &\tilde{\max} \quad cx \\ &\text{s. a.} \quad (Ax)_i \leq_f b_i, \quad i = 1,\ldots,m \\ &\qquad x_j \in \{0,1\}, \quad j = 1,2,\ldots,n \end{aligned} \tag{1.60}$$

Introduciendo la meta difusa, considerando que la tolerancia del cumplimiento de las restricciones es p_i, y usando las funciones de pertenencia lineales de (1.60) se obtiene:

$$\begin{aligned} &\max \quad \lambda \\ &\text{s.a.} \quad \lambda \leq \mu_i(x) = 1 - [(Ax)_i - b_i]/p_i \quad \forall i \\ &\qquad x_j \in \{0,1\}, \quad \forall j \quad y \quad \lambda \in [0,1] \end{aligned} \tag{1.61}$$

Por razones computacionales Zimmermann y Pollatschek reformulan (1.61) como:

$$\begin{aligned} &\text{máx} \{0, \text{mín}(1, 1+\alpha)\} \\ &\text{s.a.} \quad x_j \in \{0,1\}, \quad \forall j \end{aligned} \tag{1.62}$$

donde: $\alpha = \max_x \min_i (b'_i - (Ax)'_i)$, $b'_i = b_i/p_i$ y $(Ax)'_i = (Ax)_i/p_i$.

Para resolver (1.62), Lai y Hwang [107] proponen el uso del método de ramificación y acotación.

Por otro lado, la propuesta de Herrera y Verdegay [80], a difierencia de la anterior, consiste en el uso de funciones de pertenencia no lineales en la definición de las restricciones difusa, y la solución difusa obtienen a partir de la solución obtenida de los problemas paramétricos generados con los α-cortes.

En la versión difusa de la *programación binaria mixta* cabe destacar el trabajo de Mizunuma y Watada [137] quienes, sobre el problema de tipo clásico

$$\begin{aligned} \min \quad & z = \sum_{j=i}^{n} c_j x_j \\ \text{s.a.} \quad & \sum_{j-1}^{n} a_{ij} x_j - b_i m_i \le d_i, \; i = 1,\ldots,m \qquad (1.63) \\ & \sum_{i=1}^{m} m_i = r \\ & x_j \ge 0, \; m_i \in \{0,1\}, \; i = 1,\ldots,m; \; j = 1,\ldots,n \end{aligned}$$

introducen la difusidad en los parámetros b_i y d_i, en el sentido de admitir la tolerancia en el cumplimiento de las restricciones y la tolerancia en alcanzar la meta. Por tanto, la versión difusa de (1.63) tiene la forma del modelo siguiente:

$$\begin{aligned} \tilde{\min} \quad & z = \sum_{j=i}^{n} c_j x_j \\ \text{s.a.} \quad & \sum_{j=1}^{n} a_{ij} x_j \le_f b_i m_i + d_i, \; i = 1,\ldots,m \qquad (1.64) \\ & \sum_{i=1}^{m} m_i = r \\ & x_j \ge 0, \; m_i \in \{0,1\}, \; i = 1,\ldots,m; \; j = 1,\ldots,n \end{aligned}$$

que, usando el método de la sección 1.3.2.1, se obtiene el siguiente problema de programación binaria mixta clásico:

$$\begin{aligned} \tilde{\min} \quad & z = \sum_{j=i}^{n} c_j x_j \\ \text{s.a.} \quad & \sum_{j=1}^{n} a_{ij} x_j \le_f b_i m_i + d_i, \; i = 1,\ldots,m \qquad (1.65) \\ & \sum_{i=1}^{m} m_i = r \\ & x_j \ge 0, \; m_i \in \{0,1\}, \; i = 1,\ldots,m; \; j = 1,\ldots,n \end{aligned}$$

donde: Z_0 y Z_1-Z_0 corresponden a b_0 y t_0 de (1.14) y $b_i^1 m_i + d_i^1 \quad y \quad b_i^0 m_i + d_i^0 - (b_i^1 m_i + d_i^1)$ corresponden a b_i y t_i de (1.15).

El problema (1.65) se puede resolverse mediante los métodos clásicos, como por ejemplo el método de ramificación y acotación. Sin embargo debido a que este método

conduce a una explosión combinatorial, Mizunuma y Watada [137] proponen el uso de un algoritmo genético con la codificación binaria.

1.4.3 Otras extensiones y aplicaciones de la PLD

1.4.3.1 Programación lineal difusa interactiva

Desde que apareció la investigación de las operaciones (IO) como un conjunto de herramientas que permiten resolver problemas de decisión, la necesidad de la interacción entre el especialista de la IO y el decisor para resolver los problemas de decisión ha crecido paulatinamente hasta alcanzar el grado de indispensable en el momento en que se incluyó la teoría difusa en este campo.

Como se ha visto directamente en los métodos presentados, y como ha ocurrido a lo largo del desarrollo de la PLD, siempre ha estado presente el decisor, por ejemplo, en la obtención de las funciones de pertenencia, en la evaluación de los resultados, etc. Sin embargo, una simple interacción no es suficiente para aprovechar todo el potencial de los diversos métodos que proporciona la PL difusa y no difusa en la solución de problemas reales de decisión.

La PLD interactiva en realidad puede ser considerado como SGBM de un (SSD) orientado a cualquier tipo de problemas y a usuarios en general; integra tanto las técnicas difusas con todas sus variantes como las no difusas.

La PLD interactiva ha sido estudiada desde 1980, inicialmente por Baptistella y Otero [8] seguidos por Fabian, Ciobanu y Stoica [61], Ollero, Aracil y Camacho [152], Sakawa y Yano [173], Seo y Sakawa [178], Sakawa [165], Slowinski [182], Werners [208,209] y Zimmermann [224]. Posteriormente, incorporando las recientes técnicas de la PLD, Lai y Hwang [105,108] y Cadenas y Verdegay [29] consolidaron la PLD interactiva como sistemas soporte de decisión. Sobre todo Cadenas y Verdegay presentan el PROBO, un diseño de un SSD para problemas de optimización lineal difusa.

La PLD interactiva no es más que la interrelación de los diferente métodos difusos o no difusos de la PL, para ser utilizados en la solución de problemas reales en forma conjunta por el especialista en PL y el decisor, con la finalidad de obtener mejores decisiones, de tal manera que el sistema de producción en consideración tenga la característica de alta productividad, en el sentido de que el sistema aproveche la máxima capacidad de los recursos con el mínimo desperdicio, mínimo inventario, sin tiempos muertos, etc.

La PLD interactiva frente a un problema real, como primer paso, considera un modelo de PL en el sentido clásico; si la solución obtenida es adecuada para el decisor, y la acepta como solución final, el proceso termina. La otra posibilidad es que la primera solución no sea adecuada para el decisor, esto puede deberse a que los datos tomados como exactos no lo son en realidad, por tanto se debe formular un modelo difuso en el que a su vez las funciones de pertenencias pueden ser proporcionadas o no por el decisor. En función a estas características hay que optar por alguna de las aproximaciones presentadas en la sección (1.3). Así, en función al criterio del decisor, se adecúa el problema hasta considerar un problema completamente difuso, y al final el decisor tendrá una cantidad suficiente de alternativas entre las que elegir.

El algoritmo detallado de la PLD interactiva puede encontrarse en [107].

La PLD interactiva también se ha extendido a casos especiales, así Sakawa y Yano [172,174,213] a la programación fraccional, y Nagata, Yamaguchi y Kono [142] en los problemas de transporte utilizan la PL interactiva.

1.4.3.2 Programación estocástica difusa

La programación estocástica, introducida en la década 50, es aquella forma de PL cuyos parámetros están dados mediante distribuciones de probabilidad. Desde su aparición hasta la actualidad se ha desarrollado ampliamente, tanto en la diversidad de sus métodos de solución como en la variedad de sus aplicaciones. Dos excelentes resúmenes son los presentados por Stancu-Minasian [185] y Wets [210].

Desde que Zadeh [217] desarrollara la teoría de la posibilidad como una generalización de la teoría de la probabilidad asociada con los conjuntos difusos, la extensión de la PLD en la programación estocástica no se hizo esperar. Fue Luhandjula [125,127,128] el primero en introducir la teoría difusa en la programación estocástica, posteriormente se publicaron una serie de artículos como son los de Benoit [14], Buckley [24,25,26], Fedrizzi y Fullér [64], Hussein [86], Inuiguchi e Ichihashi [90],Inuiguchi, Ichihashi y Kume [92], Lai y Hwang [106,109,110], Masaaki [135], Wang y Zhong [207,220], Yokoyama, Ohta y Yamaguchi [215], Zhong, Zhang y Wang [221], etc. Entre los más recientes tenemos a Inuiguchi y Sakawa [93,94], Liu e Iwamura [121, 122] y Hulsurkar, Biswal y Sinha [85] entre otros.

La PL posibilística (estocástica) difusa es tan amplia como la propia PLD, siendo objeto de atención por parte de muchos autores, como se ha visto en el párrafo anterior. Debido a ello, no se ha tratado aquí pues requiere especial dedicación, pudiéndose recurrir a las referencias indicadas y al resumen de la gran mayoría de los trabajos, que podemos encontrar en [107] y también en [183], para tener una panorámica completa de modelos, métodos y aplicaciones.

1.4.3.3 Aplicaciones específicas de la PLD

Aquí reseñamos algunas aplicaciones importantes de la PLD que no están incluidas en los casos especiales anteriores.

Comenzamos con las aplicaciones propuestas por Chang, Bill y Hopkins [40], quienes proponen esquemas para modelamiento de problemas complejos, incluyendo en ellos las técnicas difusas para incrementar la flexibilidad, y luego presentan como ilustraciones los problemas de la planificación del uso de las tierras, y el sistema de tratamiento de las represas de agua en EE.UU. Un trabajo parecido en la parte aplicativa presenta Slowinski [181, 182] al utilizar las técnicas de la programación difusa multiobjetivo en el sistema de desarrollo de la planificación de la distribución del agua. Asimismo, Yamaguchi y Kono [211] presentan la aplicación de la PLD

multiobjetivo en la planificación del cultivo de tomates y pepinos. Mizunuma y Watada [137] utilizan la programación binaria mixta difusa en la distribución de los recursos.

Nagata, Yamaguchi y Kono [142] estudian un sistema de producción, almacenamiento y transporte en el que la producción está asociada con la demanda, pero esta demanda es fundamentalmente una predicción y por tanto, inexacta o bien una cantidad vaga, por lo que los autores aplican los métodos de la PLD en todo un sistema completo.

Campos [32] utiliza los modelos y los métodos de la PLD en el juego de la "suma-cero" entre dos personas. Para cada jugador establece un problema de PLD donde, durante el proceso de solución, considera distintos tipos de ordenación de los números difusos para así obtener diferentes tipos de soluciones. El método de solución que propone es una natural generalización de los métodos de solución convencionales de los juegos clásicos.

Aplicaciones de la PLD publicadas últimamente y dignas de citar son las de Chang, Chen y Wang [41], Lee y Wen [117], Ravi y Reddy [158] y Teng y Tzeng [191]. Chang, Chen y Wang después de plantear un método de solución de un problema de PL entera mixta multiobjetivo cuyos parámetros son intervalos difusos, presentan su aplicación en la planificación de los sistemas de tratamiento de desechos sólidos. Lee y Wen aplican la PL con metas difusas en el tratamiento de la calidad del agua en la cuenca de un río. Ravi y Reddy aplican la programación fraccional difusa con metas en la planificación de las operaciones en una refinería. Teng y Tzeng aplican la PLD multiobjetivo en la selección de proyectos de inversión en transporte. En un artículo recientemente publicado, Cadenas y Verdegay [28] proponen el uso de la PLD multiobjetivo para la gestión de recursos agrícolas.

1.4.4 Extensiones y aplicaciones recientes

Como se ha visto, tanto las aplicaciones como los métodos desarrollados se han dedicado a la PLD en la parte simple (pequeña escala) y a lo sumo sólo se ha llegado a la programación multiobjetivo, prestándose escasa atención a la PL en gran escala o a importantes problemas, como el de la mochila. En esta sección nos dedicamos a la version difusa de la PL en gran escala y el problema de la mochila.

1.4.4.1 Programación lineal difusa en gran escala

Como es bien conocido la PL constituye una herramienta fundamental en la toma de decisiones. Pero, las decisiones dependen de una inmensa cantidad de factores, que modelados matemáticamente, incluyen muchísimas variables de decisión. Un importante problema de este tipo es el denominado problema de *PL en gran escala* (PL-GE). Afortunadamente, en la gran mayoria de situaciones reales, con el incremento del número de variables los modelos se hacen menos densos, es decir, aumentan el número de coeficientes nulos, y sobre todo, presentan una estructura muy especial. Tienen particular importancias, tanto porque se presentan en muchas situaciones como por ser suceptibles de ser adaptados a los casos simples, aquellos modelos denominados con estructura angular de bloques, algo parecidos en su estructura a la de los problemas de transporte. El estudio de modelos con estas características reciben el nombre de *programación lineal en gran escala con estructura angular de bloques* (PLA-GE), y tiene la forma de:

$$\begin{aligned} \min \quad & cx = c_1 x_1 + \ldots + c_p x_p \\ \text{s.a.} \quad & Ax = A_1 x_1 + \ldots + A_p x_p \le b, \\ & B_1 x_1 \qquad\qquad \le b_1, \\ & \ddots \\ & \qquad\qquad B_1 x_1 \le b_1, \\ & x_i \ge 0, \quad i = 1,\ldots, p, \end{aligned} \tag{1.66}$$

donde: c_i es un vector fila r_i-dimensional y x_i es un vector columna r_i-dimensional de variables de decisión, (i=1,...,p). Las restricciones $Ax = A_1x_1+\ldots+A_px_p \le b$ son un conjunto de s restricciones que incluye todas las variables, es decir, b es un vector columna s-

dimensional y cada matriz A_i es de dimensión s x r_i, mientras que $B_i x_i \leq b_i$ (i=1,...,p) son bloques de t_i restricciones, es decir, cada b_i es un vector columna t_i-dimensional y cada matriz B_i es de dimensión t_i x r_i.

Para resolver un problema PLA-GE, Dantzig y Wolfe [46] propusieron el *algoritmo de descomposición*; algoritmo que se utiliza tanto para problemas uni y multi objetivo.

Al igual que en el caso simple, es natural pensar que en situaciones reales los parámetros de la PL-GE no siempre son conocidos con exactitud. Esto supone que en muchos casos se tendrá un problema de PL-GE difusa (PLD-GE), para el cual se debe tener los métodos de solución. Desde el punto de vista de las aplicaciones, las de mayor importancia son los PLA-GE y consecuentemente tiene mayor importancia PLA-GE difusa. El proceso de introducción de la teoria difusa en PLA-GE ha tenido la misma evolución que el caso simple, es decir, los primeros trabajos han considerado la meta difusa y las restricciones difusas (en el sentido de que se admiten violaciones tanto a las restricciones como a las metas), luego se ha abordado el PLA-GE multiobjetivo difuso, y finalmente, la PLA-GE con números difusos como parámetros.

El grupo pionero en este tema está encabezado por M. Sakawa, de la Universidad de Hiroshima, Japón; grupo que en una serie de artículos publicados entre 1993 a 1998 abarca la gran parte de la PLA-GE difusa.

En el primer artículo Sakawa, Inuiguchi y Sawada [166] estudian la PLA-GE difusa, entendida la difusidad como la aceptación del incumplimiento de las restricciones. En los siguientes tres artículos Sakawa y Sawada [171,176] y Sakawa, Inuiguchi y Sawada [167] proponen un algoritmo interactivo para el caso de PLA-GE multiobjetivo en el que se admite el incumplimiento de las restricciones y metas. Luego, Sakawa, Kato y Mizouchi [170] y Sakawa y Kato [168] estudian la PLA-GE difusa con números difusos, y finalmente, Kato, Sakawa y Ikegame [100] estudian PLA-GE binaria difusa con parámetros difusos y Sakawa y Kato [169], la PL fraccional difusa con estructura angular de bloques.

Iniciamos con la revisión de la más simple versión difusa del problema (1.66), como es el caso de la flexibilidad en el cumplimiento de las restricciones y la meta.

Consideremos que la meta ideal para el decisor es que el costo sea menor o igual que z_0^1, pero es aceptable un costo mínimo menor o igual que z_0^0, con el grado de aceptación (función de pertenencia de la función objetivo) siguiente:

$$\mu_0(cx)=\begin{cases}1, & cx \le z_0^1 \\ \alpha_0 cx+\beta_0 & z_0^1 \le cx \le z_0^0, \\ 0, & cx \ge z_0^0.\end{cases} \qquad (1.67)$$

De la misma manera, para las s primeras restricciones se tiene las funciones de pertenencia siguientes:

$$\mu_j((Ax)_j)=\begin{cases}1, & (Ax)_j \le z_j^1 \\ \alpha_j(Ax)_j+\beta_j & z_j^1 \le (Ax)_j \le z_j^0, \\ 0, & (Ax)_j \ge z_j^0.\end{cases} \qquad (1.68)$$

$$j=1,\ldots,s$$

en (1.67) y (1.68):

$$\alpha_j=\frac{1}{z_j^1-z_j^0} \quad \text{y} \quad \beta_j=\frac{z_j^0}{z_j^0-z_j^1}, \quad j=0,1,\ldots,s. \qquad (1.69)$$

Empleando una decisión convexa difusa, Sakawa, Inuiguchi y Sawada [166] formulan la versión difusa del problema (1.66) en el cual son difusas tanto las metas como aquellas restricciones que contienen todas las variables, de la siguiente manera:

$$\begin{aligned}&\max \quad w_0\mu_0(cx)+\sum_{j=1}^{s} w_j\mu_j((Ax)_j), \\ &\text{s.a.} \quad B_i x_i \le b_i, i=1,\ldots,p, \qquad (1.70) \\ &\qquad x_i \ge 0, i=1,\ldots,p, \\ &\qquad cx \le z_0^0, \\ &\qquad (Ax)_j \le z_j^0, \quad j=1,\ldots,s,\end{aligned}$$

donde $w_j \geq 0$ (j=1,...,s) y $\Sigma w_j = 1$.

Usando (1.67) y (1.68), el modelo (1.70) se puede desglosar en p subproblemas simples (pequeña escala) independientes, uno para cada bloque de restricción, obteniéndose así el siguiente:

$$\left.\begin{array}{l} \max \quad w_0\alpha_0 c_i x_i + \sum_{j=1}^{s} w_j \alpha_j (A_i x_i)_j \\ \text{s.a.} \quad B_i x_i \leq b_i, \\ \qquad x_i \geq 0. \end{array}\right\} \quad i = 1,\ldots,p \qquad (1.71)$$

Sakawa, Inuiguchi y Sawada [166] demuestran que si las p soluciones óptimas x'_i (i=1,..,p) de (1.71) satisfacen las siguientes desigualdades:

$$z_0^1 \leq \sum_{i=1}^{p} c_i x'_i \leq z_0^0, \qquad (1.72)$$

$$z_j^1 \leq (\sum_{i=1}^{p} A_i x'_i)_j \leq z_j^0, \quad j = 1,2,\ldots,s \qquad (1.73)$$

entonces $x^*=(x'_1{}^T, x'_1{}^T,\ldots, x'_1{}^T)^T$ es la solución óptima de (1.70).

En el caso de que x'_i (i=1,...,p) no satisfaga las desigualdades (1.72) y (1.73), la solución del problema (1.70) se tendrá que encontrar por otro camino. Ese otro camino también es trazado por Sakawa, Inuiguchi y Sawada [166]. Ellos demuestran que si $z^1{}_0$ y $z^1{}_j$ son soluciones óptimas de

$$\begin{array}{l} \min \quad \sum_{i=1}^{p} c_i x_i \\ \text{s.a.} \quad B_i x_i \leq b_i, i = 1,\ldots,p \\ \qquad x_i \geq 0, i = 1,\ldots,p \end{array} \qquad (1.74)$$

y

$$\begin{array}{l} \min \quad \sum_{i=1}^{p} (A_i x_i)_j \\ \text{s.a.} \quad B_i x_i \leq b_i, i = 1,\ldots,p \\ \qquad x_i \geq 0, i = 1,\ldots,p \end{array} \qquad (1.75)$$

respectivamente, entonces el problema (1.70) es equivalente al problema siguiente:

$$\begin{aligned} \min \quad & \sum_{i=1}^{p} c_i x_i \\ \text{s.a.} \quad & \sum_{i=1}^{p} (A_i x_i)_j \le z_j^0,\ j = 1,2,\ldots,s \qquad (1.76) \\ & B_i x_i \le b_i,\ i = 1,\ldots,p \\ & x_i \ge 0,\ i = 1,\ldots,p. \end{aligned}$$

y éste es un problema de PLA-GE cuya solución se puede obtener mediante el algoritmo de descomposición de Dantzig y Wolfe.

Finalmente, para calcular los pesos w_j (j=0,1,...,s) que se han utilizado en (1.70) se necesita que el decisor proporcione un valor ideal de entrada tanto para la meta como para las restricciones difusas, notado $t_j < z^0_j$ (j=0,1,...,s). Luego

$$w_j = \frac{\overline{\mu_j(t_j)}}{\sum_{i=1}^{s} \frac{1}{\overline{u_j(t_i)}}},\quad j = 0,1,\ldots,s. \qquad (1.77)$$

Todo el procedimiento para resolver un problema de PL en gran escala con estructura angular de bloques (PLA-GE) con meta y restricciones (no bloques) difusas quedaría resumido de la siguiente manera:

*Paso*1: Desglosar en problemas (1.74) y (1.75) y resolverlos transformando en subproblemas. Determinar z_0^1 y z^1_j, (j=1,...,s), soluciones de (1.74) y (1.75) respectivamente.

Paso 2: Presentar z_0^1 y z^1_j, (j=1,...,s) al decisor para que con ellos pueda proporcionar los valores ideales de entrada t_i.

Paso 3: Calcular los pesos w_j (j=0,1,...,s) según (1.77).

Paso 4: Resolver los p problemas de PL a pequeña escala en (1.71) y encontrar la solución óptima $x^*=(x'^{1T}, x'^{1T},\ldots, x'^{1T})^T$ para el problema (1.70).

Paso 5: Analizar si x^* satisface las desigualdades (1.72) y (1.73). Si es afirmativo, x^* es la solución del problema PLA-GE con la meta y restricciones difusas.

Paso 6: Si x^* no satisface (1.72) y (1.73), resolver (1.76) mediante el algoritmo de descomposición de Dantzig y Wolfe y obtener $x^{@}$ solución optima de (1.76), que también es solución satisfactoria del problema PLA-GE con meta y restricciones difusas.

Ahora pasamos a la revisión de la utilización de PLD en la *programación lineal multiobjetivo en gran escala con estructura angular de bloques* (PLMOA-GE). En forma similar a la propuesta de Zimmermann, se ha propuesto hacer el uso de las técnicas difusas en la solución PLMOA-GE.

Consideremos el problema

$$\begin{aligned}
&\min \quad c_1 x = c_{11}x_1 + \ldots + c_{1p}x_p \\
&\min \quad c_2 x = c_{21}x_1 + \ldots + c_{2p}x_p \\
&\qquad \vdots\vdots\vdots\vdots \\
&\min \quad c_k x = c_{k1}x_1 + \ldots + c_{kp}x_p \\
&\text{s.a.} \quad Ax = A_1x_1 + \ldots + A_px_p \le b_0, \qquad (1.78) \\
&\qquad\qquad B_1x_1 \qquad\qquad\quad \le b_1, \\
&\qquad\qquad\qquad\quad \ddots \\
&\qquad\qquad\qquad\qquad\quad B_1x_1 \le b_p, \\
&\qquad x_i \ge 0, \quad i = 1,\ldots, p,
\end{aligned}$$

donde los c_{ij} son r_i-dimensional vectores fila y los demás igual que en (1.66).

Debido a que en muchos problemas multiobjetivo no existe una solución completa que minimice simultáneamente todas las funciones objetivo, se usa el muy conocido concepto de *solución Pareto óptima*. Si en (1.78) notamos por X al conjunto factible, $x^* \in X$ es una *solución Pareto óptima*, si y sólo si no existe otro $x \in X$ tal que $c_i x \le c_i x^*$ para todo $i=1,\ldots,k$ y $c_j x < c_j x^*$ para algún j. La *solución Pareto óptima débil* se define en forma similar exigiendo la desigualdad estricta para todas las funciones objetivo.

Para resolver (1.78), mediante métodos difusos, Sakawa y Sawada [171,176] utilizan metas difusas (propuestas por el decisor) definidas medinate funciones de pertenencia de la forma (1.67) e introducen una forma de decisión difusa en la que utilizan μ_j^* (j=1,...,k), valores que reflejan un nivel de cumplimiento deseado por el

decisor para cada función objetivo (este valor inicial lo proporciona el decisor). En consecuencia, una solución Pareto óptima de (1.78) deberá tener una función de pertenencia muy cercana a dichos valores deseados, es decir, se debe resolver el siguiente problema:

$$\min_{x\in X} \max_{j=1,..,k} \{\mu_j^* - \mu_j(c_j x)\}. \tag{1.79}$$

o equivalentemente

$$\begin{aligned} &\min \quad \lambda \\ &\text{s.a.} \quad \mu_j^* - \mu_j(c_j x) \le \lambda, \quad j = 1,...,k \\ &\qquad x \in X. \end{aligned} \tag{1.80}$$

denominado problema mini-max.

Sakawa y Sawada demuestran que la solución óptima x^* de (1.80), si es única, es solución Pareto óptima del problema inicial (1.78). Si no es única, es una solución Pareto óptima débil, pero a partir de dicha solución se puede obtener una solución Pareto óptima resolviendo el siguiente problema test:

$$\begin{aligned} &\max \quad \sum_{j=1}^{k} \varepsilon_j \\ &\text{s.a.} \quad c_j x + \varepsilon_j = c_j x^*, \quad j = 1,...,k \\ &\qquad x \in X, \quad \varepsilon_j \ge 0,\ j = 1,...,k \end{aligned} \tag{1.81}$$

Si $(x^{\cdot},\varepsilon^{\cdot})$ es la solución óptima de (1.81), entonces $x^{\cdot}$ es la solución Pareto óptima de (1.78).

Los problemas (1.80) y (1.81) se resuelven mediante el algoritmo de descomposición de Dantzig-Wolfe.

En resumen, el algoritmo interactivo [171,176] es el siguiente:

Paso 0: Obtener el valor mínimo individual $z_j^{\min}$ y valor máximo individual $z_j^{\max}$ de cada función objetivo con las restricciones del problema inicial (1.78).

Paso 1: El decisor determina las funciones de pertenencia que expresan las metas difusas para cada función objetivo considerando el intervalo cerrado $[z_j^{\min}, z_j^{\max}]$.

Paso 2: El decisor proporciona la referencia inicial μ_j^* (j=1,...,k).

Paso 3: Obtener una solución óptima de Pareto resolviendo el problema (1.80) y el correspondiente problema test (1.81).

Paso 4: Si el decisor no se siente satisfecho con la solución obtenida en el paso 3, modificar los valores de la referencia inicial y luego retornar al paso 3.

Si se opta por resolver el problema (1.78) utilizando el operador min, [167] el procedimiento es equivalente, con la excepción del paso 2, y el problema a resolver es

$$\begin{aligned} &\max \quad \lambda \\ &\text{s.a.} \quad \mu_j(c_j x)) \geq \lambda, \quad j = 1,...,k \\ &\qquad x \in X. \end{aligned} \tag{1.82}$$

en lugar del problema (1.80).

Finalmente, revisamos *La PLAD-GE multiobjetivo con parámetros difusos*. Esto se origina cuando tanto los parámetros como los costos unitarios, los coeficientes tecnológicos y los recursos en general, no se conocen con exactitud, es decir, el problema es completamente difuso. El modelo asociado a este problema se expresa de la siguiente manera:

$$\begin{aligned} &\min \quad \tilde{c}_1 x = \tilde{c}_{11} x_1 + ... + \tilde{c}_{1p} x_p \\ &\min \quad \tilde{c}_2 x = \tilde{c}_{21} x_1 + ... + \tilde{c}_{2p} x_p \\ &\qquad :::: \\ &\min \quad \tilde{c}_k x = \tilde{c}_{k1} x_1 + ... + \tilde{c}_{kp} x_p \\ &\text{s.a.} \quad \tilde{A} x = \tilde{A}_1 x_1 + ... + \tilde{A}_p x_p \leq_f \tilde{h}, \\ &\qquad \tilde{B}_1 x_1 \qquad\qquad \leq_f \tilde{b}_1, \\ &\qquad\qquad\qquad \ddots \\ &\qquad\qquad\qquad\quad \tilde{B}_1 x_1 \leq_f \tilde{b}_p, \\ &x_i \geq 0, \quad i = 1,..., p, \end{aligned} \tag{1.83}$$

donde todos los coeficientes son como en (1.78) con la diferencia de que en (1.83) los números en los vectores, matrices y constantes en general son difusos.

En los métodos de solución que plantean Sakawa y Kato [168] y Sakawa, Kato y Mizouchi [170] los números difusos que utilizan son los números introducidos por Dubois y Prade [55].

Al igual que en el caso de la PLD uniobjetivo, para encontrar una solución aproximada de (1.83), se transforma en un problema clásico utilizando los α-cortes. El conjunto de todos los α-cortes en todos los coeficientes lo representamos de la siguiente manera:

$$(\widetilde{A},\widetilde{B},\widetilde{b},\widetilde{c})_\alpha = \left\{ \begin{array}{c} (A,B,b,c) / \mu_{\widetilde{a}_{ij}^s}(a_{ij}^s) \geq \alpha, \mu_{\widetilde{b}_{hj}^s}(b_{hj}^s) \geq \alpha, \mu_{\widetilde{b}'^{s'}_{h'}}(b'^{s'}_{h'}) \geq \alpha, \\ \mu_{\widetilde{c}_j^{ts}}(c_j^{ts}) \geq \alpha, i = 1,...,t_0,\ j = 1,...,r_s, s = 1,...,p \\ h = 1,...,t_s, h' = 1,...,t_{s'}, s' = 0,...,p, t = 1,...,p \end{array} \right\} \qquad (1.84)$$

Si elejimos un elemento del conjunto (1.84) y sustitimos en lugar de los coeficientes del modelo (1.84), se obtiene el modelo no difuso siguiente

$$\begin{array}{ll} \min & c_1 x = c_{11}x_1 + ... + c_{1p}x_p \\ \min & c_2 x = c_{21}x_1 + ... + c_{2p}x_p \\ & :::: \\ \min & c_k x = c_{k1}x_1 + ... + c_{kp}x_p \\ \text{s.a.} & Ax = A_1x_1 + ... + A_px_p \leq b_0, \\ & B_1x_1 \qquad\qquad \leq b_1, \\ & \qquad\ddots \\ & \qquad\qquad B_1x_1 \leq b_p, \\ & x_i \geq 0, \quad i = 1,...,p; \\ & (A,B,b,c) \in (\widetilde{A},\widetilde{B},\widetilde{b},\widetilde{c})_\alpha . \end{array} \qquad (1.85)$$

que denominamos problema α-PLDA-GE.

El modelo (1.85) es del tipo clásico, pero debido a que en realidad los coeficientes no son números específicos sino variables, no se puede aplicar el concepto de solución Pareto óptima, por lo que es necesario generalizar este concepto para el caso como solución α-Pareto óptima. Un $x^* \in X(A^*,B^*,b^*)$ se denomina *solución α-Pareto óptima* de α-PLDA-GE

(1.85) si y sólo si no existe otra $x \in X(A,B,b)$, tal que $c_i x \leq c^*_i x^*$, $i=1,...,k$, y si se cumple la desigualdad estricta para algún i.

Para generar la solución candidata a ser α-Pareto óptima de (1.85), antes, al igual que para la PLDA-GE multiobjetivo, el decisor debe proporcionar el nivel de referencia inicial z'_i, $i=1,...,k$. La solución α-Pareto óptima será aquélla que esté lo más cercanamente posible a dichos niveles de referencia, es decir, que sea la solución del siguiente problema:

$$\begin{aligned} \min \quad & \max_{i=1,...,k}(c_i x - z'_i) \\ \text{s.a.} \quad & x \in X(A,B,b), \\ & (A,B,b,c) \in (\widetilde{A},\widetilde{B},\widetilde{b},\widetilde{c})_\alpha \end{aligned} \tag{1.86}$$

o equivalentemente

$$\begin{aligned} \min \quad & \lambda \\ \text{s.a.} \quad & (c_i x - z'_i) \leq \lambda,, \quad i=1,...,k \\ & x \in X(A,B,b), \\ & (A,B,b,c) \in (\widetilde{A},\widetilde{B},\widetilde{b},\widetilde{c})_\alpha. \end{aligned} \tag{1.87}$$

Este problema (1.86), debido a que los parámetros son considerados como variables de decisión tiene las restricciones no lineales. Pero dichos parámetros tienen los intervalos de variación que son los respectivos α-cortes, intervalos cerrados de la forma:

$$\left[c^L_{i\alpha}, c^R_{i\alpha}\right],\ \left[A^L_\alpha, A^R_\alpha\right],\ \left[B^L_{j\alpha}, B^R_{j\alpha}\right] \text{ y } \left[b^L_{j\alpha}, b^R_{j\alpha}\right],$$

para los parámetros c_i, A, B_j y b_i respectivamente.

Sakawa y Sawada demuestran que la solución candidata a ser α-Pareto óptima de (1.86) o (1.87), se puede obtener resolviendo el problema obtenido de (1.86) tomando los coeficientes de los términos del primer miembro, los extremos izquierdo de los intervalos (α-cortes), y los extremos derecho para los parámetros de los términos del segundo miembro, siguiente:

$$
\begin{aligned}
&\min \quad \lambda \\
&\text{s.a.} \quad c_{1\alpha}^{L} = c_{11\alpha}^{L} x_1 + \ldots + c_{1p\alpha}^{L} x_p - z'_1 \le \lambda, \\
&\qquad c_{2\alpha}^{L} = c_{21\alpha}^{L} x_1 + \ldots + c_{2p\alpha}^{L} x_p - z'_2 \le \lambda, \\
&\qquad\qquad \vdots \\
&\qquad c_{k\alpha}^{L} = c_{k1\alpha}^{L} x_1 + \ldots + c_{kp\alpha}^{L} x_p - z'_k \le \lambda, \qquad (1.88) \\
&\qquad A_{\alpha}^{L} x = \; A_{1\alpha}^{L} x_1 + \ldots + A_{p\alpha}^{L} x_p \le b_{0\alpha}^{R}, \\
&\qquad\qquad B_{1\alpha}^{L} x_1 \qquad\qquad\qquad \le b_{1\alpha}^{R}, \\
&\qquad\qquad\qquad\qquad \ddots \\
&\qquad\qquad\qquad\qquad\qquad B_{1\alpha}^{L} x_1 \le b_{1\alpha}^{R}, \\
&\qquad\qquad x_j \ge 0, \quad j = 1,\ldots, p.
\end{aligned}
$$

Si x^* es la solución óptima de (1.88), candidata a solución α-Pareto óptima, la solución $x^{@}$ α-Pareto óptima de (1.87) se obtiene de la solución óptima ($x^{@}$, $\varepsilon^{@}$) del problema test:

$$
\begin{aligned}
&\max \quad \varepsilon = \sum_{i=1}^{k} \varepsilon_i \\
&\text{s.a.} \quad c_{i\alpha}^{L} x + \varepsilon_i = c_{i\alpha}^{L} x^*, \qquad (1.89) \\
&\qquad A_{\alpha}^{L} x \qquad \le b_{0\alpha}^{R}, \\
&\qquad B_{\alpha}^{L} x_j \qquad \le b_{j\alpha}^{R}, \\
&\qquad x_j \ge 0, \quad \varepsilon_i \ge 0, \quad i = 1,\ldots, k, \; j = 1,\ldots, p.
\end{aligned}
$$

Este procedimiento se puede resumir en un algoritmo cuyo paso inicial consiste en hallar el mínimo y máximo individual para cada función objetivo usando α=0 y α=1 respectivamente. Estos valores pueden servir como una referencia para elegir los niveles de referencia iniciales en el siguiente paso. Luego se determina la solución α-Pareto optima resolviendo sucesivamente los problemas (1.88) y (1.89). Y finalmente, si la solución obtenida no es de su agrado, el decisor elegirá otros niveles de referencia hasta alcanzar la solución aceptable.

Como aplicaciones del método desarrollado en esta se han desarrollado algunos trabajos. Comentamos dos casos especiales, uno de ellos se refiere a la solución de la programación fraccional difusa multiobjetivo en gran escala con estructura angular de bloques (PFADM-GE), desarrollado por Sakawa y Kato [169], y el otro, a la solución de

problemas de programación binaria difusa multiobjetivo en gran escala con estructura angular de bloques, desarrollado por Kato, Sakawa y Ikegame [100].

En ambos casos, los problemas difusos se caracterizan porque sus coeficientes son números difusos al igual que en (1.83), por tanto, todo el procedimiento de transformación al caso clásico viene a ser igual.

En la programación fraccional [169], las expresiones fraccionales de la función objetivo que pasan a ser restricciones, se pueden transformar en simples restricciones lineales. Y en la programación binaria [100], después de transformar al caso clásico se aplica un algoritmo genético para hallar la solución.

1.4.4.2 El problema de la mochila difuso

Cuando los parámetros del problema de la mochila no son conocidos exactamente, su formulación matemática podrá realizarse con parámetros difusos.

Al igual que para el problema de la mochila clásico (con parámetros exactos), para resolver el problema de la mochila difuso no es posible generalizar los métodos de la PLD que son propios de los problemas con variables no discretas. Esto ha motivado a algunos autores a buscar métodos de solución muy específicos.

Uno de los trabajos iniciales para resolver el problema de la mochila difuso ha sido realizado por Okada y Gen [150] para el problema simple, es decir, con una sola restricción. Los mismos autores en otro artículo [151] proponen un método de solución para el problema difuso multidimensonal de la mochila:

$$\begin{aligned} \max \quad & \tilde{z} = \sum_{j=1}^{n} \tilde{c}_j x_j \\ \text{s.a.} \quad & \sum_{j=1}^{n} \tilde{a}_{ij} x_j \leq^{f} \tilde{b}_i, \quad i = 1,\ldots,m \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \tag{1.90}$$

donde todos los coeficientes son números triangulares difusos positivos.

La relación de desigualdad entre los números difusos dada por Kaufmann y Gupta es utilizada por Okada y Gen en las desigualdades de las restricciones en (1.90). Según Kaufmann y Gupta, la relación de desigualdad entre dos números difusos está dada por

un número que indica el grado de desigualdad existente entre dichos números. Dicho grado está dado por:

$$P(\widetilde{a} \leq^{f} \widetilde{b}) = \frac{\int_{a_L}^{a_R} min\{\mu_a(x), \mu_B(x)\}dx}{\int_{a_L}^{a_R} u_a(x)dx} \qquad (1.91)$$

donde: $\widetilde{a} = (a_L, a_M, a_R)$ y $\widetilde{b} = (b_L, b_M, b_R)$ son números difusos triangulares, y $\widetilde{B}$ definido por la función de pertenencia $\mu_B(x) = \sup_x \{\mu_b(u) / x \leq u\}$ se denomina cota superior difusa para el número difuso $\widetilde{b}$.

Como es sabido, una terna que representa a un número triangular difuso, indica que la función de pertenencia correspondiente tiene valor cero en $(-\infty, a_L]$ y en $[a_R, \infty)$; la función de pertenencia es lineal creciente en $[a_L, a_M]$, y lineal decreciente en $[a_M, a_R]$, y tiene como máximo valor 1 en a_M . Si los tres componentes son positivos se dice que el número difuso triangular es positivo.

Okada y Gen utilizan el eficiente método del gradiente para resolver el problema (1.90). Primeramente fijan un grado mínimo α en que las desigualdades de los números difusos de las restricciones deben ser aceptadas. Este grado mínimo es dado por el decisor. Los autores citados proponen un algoritmo que se compone de dos fases. La primera fase consiste en generar un conjunto de soluciones factibles que cumplan con el grado mínimo α fijado en que las desigualdades se forman, y la segunda fase consiste en elegir la mejor solución usando para ello la misma definición del grado en que se forma la desigualdad entre dos números difusos.

Recientemente, Abboud, Sakawa e Inuiguchi [1] han propuesto un método de solución para el problema de la mochila multidimensional y multiobjetivo con metas difusas,

$$\begin{aligned} \max \quad & z_l = \sum_{j=1}^{n} k_{lj} x_j, \quad l = 1,\ldots,q \\ \text{s.a.} \quad & \sum_{j=1}^{n} r_{ij} x_j \leq b_i, \quad i = 1,\ldots,m \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \qquad (1.92)$$

donde $k_{lj} < 0$ y $r_{ij} \geq 0$.

Evidentemente, las metas difusas tienen que estar definidas mediante una función de pertenencia. El método que proponen los autores es para las funciones de pertenencia que son lineales y que son definidas por el decisor. En la definición de esta función, el decisor tendrá como referencia el máximo valor de cada función objetivo dentro de la región factible, y el cero, que viene a ser el mínimo valor posible para dichas funciones de pertenencia.

Abboud, Sakawa e Inuiguchi emplean una decisión convexa para transformar todas las funciones objetivo en una sola, es decir, el problema equivalente al problema (1.92) con metas difusas es:

$$\begin{aligned} \max \quad & \sum_{l=1}^{q} w_l \mu_{z_l}(k_l x) \\ \text{s.a.} \quad & Rx \leq b, \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \tag{1.93}$$

donde: $k_l = (k_{l1},\ldots,k_{ln})$, $u_{z_l}(.)$ es la función de pertenencia de la meta difusa l, $\sum_{j=1}^{q} w_j = 1, w_j > 0$ y $w_j = \dfrac{1/\mu_{z_l}(z_l^a)}{\sum_{l=1}^{q} 1/\mu_{z_l}(z_l^a)}$, con z_l^a el nivel de aspiración de la función objetivo l, que debe ser proporcionado por el decisor.

Utilizando la función de pertenencia lineal, de (1.93) se puede obtener un problema en la forma

$$\begin{aligned} \max \quad & cx \\ \text{s.a.} \quad & Ax \leq 1, \\ & x_j \in \{0,1\}, \quad j = 1,\ldots,n \end{aligned} \tag{1.94}$$

que es un problema clásico que puede resolverse mediante el eficiente método del gradiente.

Capítulo 2

CRITERIOS DE PARADA DIFUSOS EN LA PROGRAMACIÓN LINEAL

2.1 INTRODUCCIÓN

Los problemas de la realidad, cuando no es posible resolverlos adecuadamente usando el sentido común o la fuerza bruta, pueden ser resueltos usando diversos métodos que proporciona la ciencia. Para ello, deben ser expresados en un lenguaje adecuado que permita aplicar métodos de solución adecuados. Las expresiones del problema se denominan *modelos*. Si lo que se utiliza es el lenguaje matemático los modelos obtenidos se denominan *modelos matemáticos*.

Hasta hace dos décadas existían dos principales formas de clasificación de modelos matemáticos: discretas versus continuas y determinísticas versus estocásticas (probabilísticas). Ahora podemos clasificarlos de una forma general en modelos difusos y no difusos y en cada uno de estos grupos efectuar las clasificaciones tradicionales.

Los modelos matemáticos discretos han sido los primeros en aparecer (aunque no se denominaran así) por basarse fundamentalmente en el conjunto de los números naturales.

Los diversos métodos que se utilizan para resolver problemas que generan modelos matemáticos discretos son denominados en términos generales ***algoritmos***, si bien es cierto que no se conoce exactamente cuando se optó por esta denominación. Inicialmente, junto con la introducción de la notación arábiga de los números naturales se introdujo la denominación de *algorismo* para referirse a un determinado proceso de cálculo mediante números arábigos.

Si entendemos que un **algoritmo** es un proceso de cálculo con números naturales, tenemos que aceptar que los algoritmos han sido desarrollados desde la Antigüedad, ya que babilonios y egipcios desarrollaron tablas de multiplicación, y uno de los más famosos algoritmos conocidos es el algoritmo propuesto por Euclides (300 A. de C.), que lleva su nombre, para determinar el máximo común divisor de dos números.

También desde la Antigüedad se ha destacado la importancia de la eficiencia de los algoritmos, entendiéndose por eficiencia la rapidez con que éstos pueden resolver un problema. Actualmente, el análisis de la eficiencia de un algoritmo constituye un aspecto principal de la teoría de algoritmos. La rapidez de los algoritmos depende de diversos factores como son la magnitud de los datos, el uso de los instrumentos etc.

La necesidad de una mayor rapidez de los algoritmos ha llevado al desarrollo de gran número de investigaciones en los últimos años, para conseguir algoritmos más eficientes que los ya existentes, o para resolver nuevos problemas que han surgido.

Desde un punto de vista (matemáticamente) más formal [11, 16], *algoritmo* es un mapeo de un conjunto de entradas L a una familia de conjuntos de resultados $\Re$. Si *Alg* nota a la función algorítmica,

$$Alg: \mathrm{L} \rightarrow \Re$$

$$d \rightarrow \quad Alg(d) \in \Re$$

donde L, *Alg*(d) son subconjuntos de algún dominio X.

El conjunto *Alg*(s) no es un conjunto arbitrario sino es una sucesión de la forma

$$\{x_0, x_1, x_2 \ldots\ldots x_n \ldots\ldots\}$$

con la regla de correspondencia *f* definida por la parte iterativa del algoritmo, es decir,

$$x_n = f(x_{n-1})$$

se obtiene en la *n*-ésima iteración como resultado de aplicar el algoritmo al resultado de la iteración anterior, y $x_0 = d$ es el elemento (solución) inicial del algoritmo.

La sucesión $\{ x_n \}$ puede ser de rango finito o infinito. Si es de rango finito obviamente es convergente. En el caso de que sea una sucesión de rango infinito y no convergente, el algoritmo no tiene validez para la solución de problemas pues sólo tienen importancia aquellos algoritmos correctos que generan sucesiones convergentes.

Si un algoritmo correcto aplicado a un ejemplar de un problema admite más de un elemento inicial, es decir, si L es no unitario, todas las sucesiones obtenidas con los diferentes elementos iniciales en L deben converger a un mismo elemento. Si esto ocurre por ejemplo en algoritmos de optimización se dice que el elemento de convergencia es un óptimo global; si por el contrario, hay por lo menos dos elementos iniciales que generan

sucesiones con diferentes elementos de convergencia, entonces dichos elementos son denominados óptimos locales. En cualquier caso, a dichos elementos se llega siempre tras la verificación de determinados criterios de parada o de control.

Los criterios de parada constituyen una parte fundamental de un algoritmo, para ubicarlo veamos primero las estructura general de un algoritmo.

2.1.1 Estructura de un Algoritmo

Ya hemos visto que un algoritmo es un proceso repetitivo que genera una sucesión de elementos de acuerdo a un conjunto de instrucciones pre- escritas hasta cumplir con una condición de parada. Además requiere de un elemento inicial generador de la sucesión. De esto se desprende que en todo proceso algorítmico pueden distinguirse tres grandes etapas: inicial, iterativa y final.

Etapa Inicial: Es la etapa en que, sobre la base de alguna regla, se elige un *elemento inicial* para generar los demás valores de la sucesión en la etapa iterativa. De acuerdo a la naturaleza de los problemas y del algoritmo que se resuelve, dicho elemento inicial puede ser un elemento en particular, como por ejemplo en el algoritmo simplex, en los algoritmos de la mochila, etc. En otros casos, se utilizan reglas específicas para obtener el elemento inicial, como ocurre por ejemplo en el algoritmo simplex de transporte donde hay varias reglas distintas para obtener el elemento inicial. En estos últimos casos, el conjunto L mencionado es no unitario.

Etapa Iterativa: Es la etapa operativa principal del algoritmo que genera los elementos de la sucesión, elementos que constituyen soluciones aproximadas del problema.

Etapa final: Esta etapa, que en lo sucesivo denominaremos **criterio de parada**, es la que controla las repeticiones de la etapa iterativa. Está formada por un conjunto de reglas que indican cuando el procedimiento algorítmico debe finalizar,

razón por la que reciben el nombre de reglas de control o criterios de parada, de los que se ofrecen más detalles en la siguiente sección.

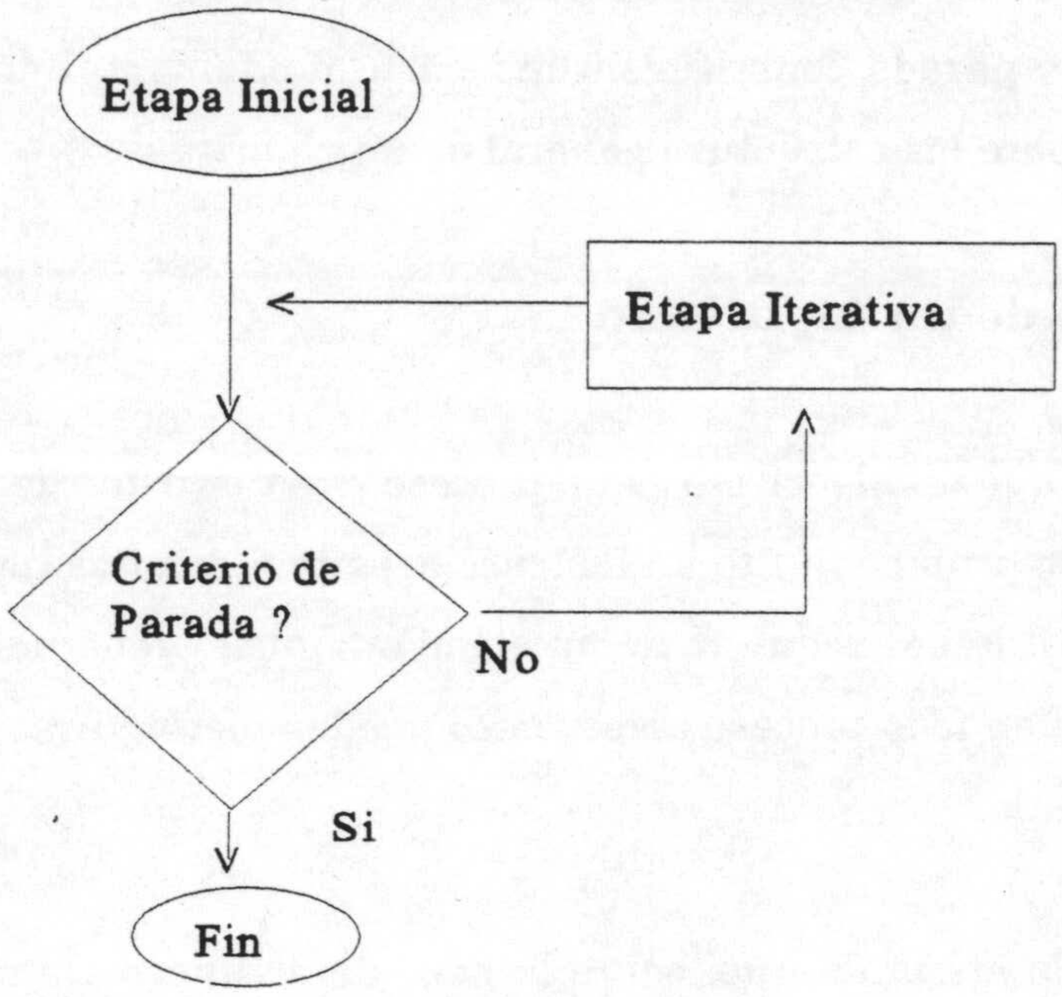


Figura 2.1. Estructura general de un algoritmo

2.1.2 Criterios de parada clásicos

Una de las características del algoritmo mencionadas en la sección anterior es la de ser finito, es decir, que se trata de un procedimiento de cálculo que debe terminar. Por otro lado, desde el punto de vista abstracto, se concibe como un procedimiento que genera una sucesión $\{ x_n \}$. Puesto que en forma general una sucesión es infinita aún en los casos de rango finito, es necesario en todo algoritmo establecer los criterios de parada.

Los criterios de parada se establecen a partir de las características teóricas del problema, del tipo de solución que se busca y del algoritmo que se utilice. Algunos de los criterios más frecuentes son:

a) Parar el proceso después de N iteraciones.

b) Parar el proceso si la diferencia entre dos elementos de la sucesión a partir de una determinada iteración es menor o igual que un valor prefijado.
Si ε es un valor prefijado, se detiene el proceso cuando

$$\|x_{N+k}-x_k\|<\varepsilon \qquad o \qquad \frac{\|x_{k+1}-x_k\|}{\|x_k\|}<\varepsilon$$

En este último caso se denomina distancia relativa.

c) Parar el proceso cuando una medida prefijada $g(x_n)$ satisface una determinada condición.
Como por ejemplo:

$$g(x_n) > c, \qquad g(x_{N+k})-g(x_k) < \varepsilon, \qquad \frac{|g(x_{k+1})-g(x_k)|}{g(x_k)} < \varepsilon.$$

En un algoritmo el criterio de parada puede estar expresado mediante una o varias reglas, ya sea en forma disyuntiva o conjuntiva.

2.1.3 Criterios de parada difusos

Los algoritmos en general son diseñados para resolver problemas, y sobre todo problemas reales. Muchos de estos problemas reales son complejos y con dimensiones grandes. En estos casos, los algoritmos -aún los que son polinomiales-, para satisfacer las condiciones de parada requieren un número de iteraciones muy grande que es posible no se puedan alcanzar en tiempo razonable; situaciones en las que aún los algoritmos más eficientes pueden ser poco útiles. La situación es peor para los problemas NP-completos, los cuales carecen de algoritmos con tiempos de ejecución polinomial. Para ellos, se han desarrollado diversos algoritmos heurísticos que permiten determinar soluciones aceptables. No obstante, esto ha supuesto descuidar la búsqueda de la posibilidad de adaptar los algoritmos exactos de tal manera que nos permitan hallar soluciones aceptables. Los algoritmos exactos, tal como están construidos, son poco flexibles ya que sus criterios de parada están basados en análisis teóricos de los problemas y algoritmos, por tanto el procedimiento de calculo hasta alcanzar la solución exacta para problemas de grandes dimensiones, puede ser incluso varios años.

En el caso de problemas de decisión, en situaciones reales, cuando no se conoce

ninguna solución, y fuera imposible obtener una solución exacta en un tiempo razonable, para el decisor será suficiente tener una solución *aceptable*. Esta categoría de aceptable respecto de una solución no exacta, sólo la puede otorgar el decisor que está al frente de un problema en particular que necesita resolver. En consecuencia, para los efectos de aplicación en problemas reales y de dimensiones arbitrarias los algoritmos deben tener una flexibilidad para adecuarse a situaciones muy particulares y también a los criterios del decisor.

Debido a que la dinámica de todo algoritmo está en sus reglas de control, la flexibilización se tiene que buscar en dichas reglas para obtener un algoritmo semejante, pero con la finalidad de que sea capaz de proporcionar soluciones aceptables en un tiempo de ejecución razonable.

Una herramienta que ha ayudado en la solución de problemas en diferentes campos en los que los datos no se conocen con exactitud es *la teoría de conjuntos difusos*. Particularmente, la programación lineal difusa ha tomado como fuente esta teoría, lo que acredita la virtualidad aplicativa de la teoría de conjuntos difusos en otros ámbitos como es el de la flexibilización de los criterios de parada, que se denominan criterios de parada difusos.

Los criterios de parada difusos fueron introducidos por primera vez en el ámbito de la programación matemática en 1995 por Verdegay [204], y luego por Herrera y Verdegay [81] en 1996, particularmente en el algoritmo simplex de la PL. En este trabajo desarrollamos esto con mayor amplitud, y lo extendemos a los algoritmos de Karmarkar y de Puntos Interiores de la PL, así como a algunos algoritmos que sirven para resolver los Problemas de la Mochila y del Viajante de Comercio.

Sin pérdida de generalidad consideremos que el criterio de parada de un algoritmo exacto es:

$$g(x_n) \geq c \tag{2.1}$$

Es evidente que una solución no óptima no cumple con la condición (2.1). Las soluciones no exactas se obtendrán cuando $g(x_n) < c$. Una solución no exacta será menos aceptable cuanto mayor sea la diferencia con c, y será muy cercana al óptimo cuando $g(x_n)$ sea muy cercano

a *c*. De acuerdo a la propuesta de Herrera y Verdegay [81], la mejor manera de expresar el nivel en que se admite el incumplimiento del criterio de parada (2.1) para alcanzar una solución aceptable, es mediante un conjunto difuso cuya función de pertenencia es de la siguiente forma:

$$\mu(x) = \begin{cases} 1 & x \geq c \\ h(x) & c - \delta < x < c \\ 0 & x \leq c - \delta \end{cases} \tag{2.2}$$

donde $h(x) \in (0,1)$ es una función continua creciente, y cuya representación gráfica se muestra en la Figura 2.2.

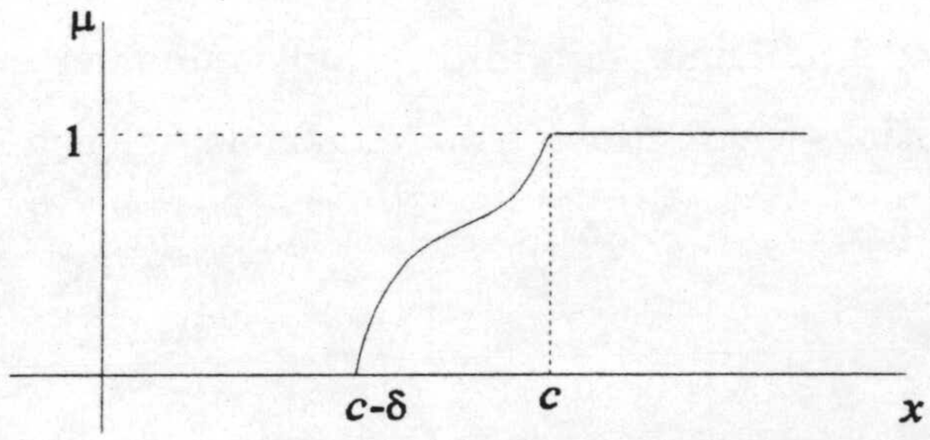


Figura 2.2. Función de pertenencia para una regla de parada difusa

El valor 1 de la función de pertenencia indica el más alto grado de aceptación, que vendría a ser la solución exacta; el valor 0 expresa el menor grado de aceptación. Naturalmente, el decisor preferirá una solución con el mayor grado posible de aceptación, para ello debe fijar el grado mínimo de aceptación que puede admitir. Si α $(0<\alpha\leq 1)$ es el grado mínimo de aceptación fijada, la regla de parada difusa correspondiente que reemplazaría a (2.1) sería (2.3).

$$\mu(g(x_n)) \geq \alpha \tag{2.3}$$

Teniendo en cuenta que la introducción del criterio de parada difuso es con la finalidad de admitir una solución no exacta del problema, el criterio de parada sería

$$h(g(x_n)) \geq \alpha \tag{2.4}$$

o equivalentemente

$$g(x_n) \geq h^{-1}(\alpha) \tag{2.5}$$

Este último se utiliza para los efectos de cálculo y en las implementaciones.

Herrera y Verdegay [81], utilizan como h una función lineal y sugieren que el valor δ utilizado en (2.2) y α de (2.3) los debe proponer el decisor. En este trabajo proponemos que el valor δ sea obtenido a partir de la etapa inicial del algoritmo o bien usando algún otro método que permita obtenerlo fácilmente, y dicho valor δ obtenido le facilitará al decisor el elegir α con mejor acierto. Asimismo, al definir la función de pertenencia en función de la naturaleza de los algoritmos y los problemas, sugerimos el uso de diversos tipos de funciones no lineales.

2.2 CRITERIOS DE PARADA DIFUSOS EN EL ALGORITMO SIMPLEX

El algoritmo simplex diseñado por G. Dantzig [47], sirve para resolver el problema de programación lineal estándar (PLE) siguiente:

$$\begin{aligned} &\max c^T x \\ &\text{s.a. } Ax=b \\ &x \geq 0, \end{aligned} \tag{2.6}$$

donde c y x (variable) son vectores columna n-dimensionales, A es una matriz mxn de rango m, b es un vector columna m-dimensional de números reales. Por la expresión del modelo, podemos suponer que se trata de un problema de producción en la que se quiere determinar la máxima utilidad.

El algoritmo simplex desarrollado para PLE es de aplicación general a todo problema de PL dado que éstas siempre es posible reducir a un PLE [156].

La región factible $F = \{ x \in \mathbb{R}^n / Ax = b, x \geq 0 \}$ se denomina polítopo y posee un número finito de vértices. Si este polítopo está acotado se denomina poliedro. Cuando la función objetivo $z = c^T x$ está acotada en F, existe un vértice con utilidad máxima [98,156].

Asociado a cada vértice x_0 de F existe una submatriz regular m-dimensional B de A denominada *base,* tal que $Bx_B = b$, x_B se denomina *solución básica factible,* y $x_0 = (x_B, 0)^T$.

Resolver el problema (2.6) es hallar un punto(vértice) factible $x^* \in F$ de máxima utilidad, es decir, $c^T x \leq c^T x^*$, $\forall x \in F$, y se denomina *solución óptima* de utilidad $z^* = c^T x^*$. Si F es vacía, (2.6) no tiene solución; y si $z = c^T x$ es no acotada en F, se dice que el problema (2.6) es no acotado o no tiene máximo.

Si se resolviera el PLE (2.6) utilizando la "fuerza bruta", se tendrían que hallar todos los vértices de F resolviendo todas las combinaciones de la forma $Bx_B = b$ que se pueden formar de $Ax = b$, y luego seleccionar el vértice de mayor utilidad. Con este método se presentan dos inconvenientes: uno, el que en el caso de un problema no acotado se llegaría a elegir una utilidad como máxima cuando en realidad no lo es, y otro, el enorme trabajo de tener que calcular el elevado número de vértices que tiene la región factible cuando n-m es muy alto.

El algoritmo simplex evita el análisis de todos los vértices pues, partiendo de un vértice inicial, recorre los vértices hasta llegar al óptimo pasando del vértice actual a otro si éste es de mayor utilidad, recorrido en el que no necesariamente tiene que visitar todos los vértices. La parte algebraica de este recorrido es muy simple: se utiliza la matriz base B asociada al vértice actual; el siguiente vértice se obtiene sustituyendo una columna de la base B por otra de la matriz A que no forma parte de B, eligiéndose la columna que ha de ser sustituida y la que la sustituye bajo la condición de que el nuevo vértice tenga la mayor utilidad de entre todos los intercambios posibles y a su vez, sea de mayor utilidad que el actual; ésta es la etapa iterativa del algoritmo. El algoritmo finaliza cuando ninguno de los vértices que restan por visitar tiene mayor utilidad que el vértice actual o cuando hay algún indicativo de que el problema es no acotado o no tiene solución.

A continuación presentamos las etapas principales del algoritmo simplex, pero antes, introduzcamos las notaciones y fórmulas que se usarán.

Sea x_0 la solución actual (vértice) y B la base correspondiente formado por las columnas { j(1),j(2),...,j(m) } = $\Im$. Las componentes $\Im$ de x_0 tienen valores correspondientes $B^{-1}b$ y el resto, nulos. Notemos por x_{i0} la j(i)-ésima componente (no nula) de x_0.

Siendo B una base, cada vector A_j ($j \notin \Im$) de la matriz A se puede expresar como combinación lineal de los elementos de la base B. En concreto, un vector de los coeficientes de la combinación es el vector columna *m*-dimensional

$$(y_{ij})_{i=1,..,m} = Y_j = B^{-1}A_j.\ j \notin \Im, \tag{2.7}$$

Por otro lado, sean:

$$\delta_j = \sum_{i=1}^{m} c_{j(i)} y_{ij} - c_j \tag{2.8}$$

y

$$\theta_0 = min\left\{\theta_i = \frac{x_{i0}}{y_{ij}},\quad i = 1,...,m,\quad y_{ij} > 0\right\} \tag{2.9}$$

Etapas del algoritmo simplex:

Etapa inicial: Encontrar un punto extremo (vértice) inicial con base B.

Etapa iterativa:

a) Sea **j'** tal que $\delta_{j'} = \min\{ \delta_j \ / \ \delta_j > 0, j \notin \Im \}$, seleccione la **j'**-ésima columna como la nueva columna básica que entra.

b) Sea **j(k)** tal que $\theta_0 = \theta_{j(k)}$, (calculados de acuerdo a (2.9)), seleccione la **j(k)**-ésima columna para ser sustituida por la nueva **j'**-ésima columna base seleccionada en (a) para formar la nueva base B'.

c) Determinar el nuevo vértice x_0'.

Criterios de Parada.

a) Si $Y_j \leq 0$, detener el proceso, pues el problema no es acotado; en caso contrario,

b) Si $\delta_j \geq 0\ \forall\ j \notin \Im$, detener el proceso pues el vértice actual es la solución óptima.

De acuerdo a estos criterios, el algoritmo se detiene cuando determina que el problema es no acotado o cuando alcanza una solución exacta. En problemas de PL de grandes dimensiones, para alcanzar este resultado pueden requerirse muchas iteraciones con

tiempos de ejecución muy prolongados. Más aún, si es no acotado no proporcionará ningún resultado aunque existan soluciones aceptables.

En estas situaciones, es posible admitir una solución no óptima, pero cercana a la óptima, y determinable en un tiempo de cálculo razonable que en general llamaremos *solución aceptable.* Herrera y Verdegay [81] proponen que para lograr una solución aceptable se puede modificar la condición de parada. La condición para obtener la solución exacta "δ_j no negativa" debe ser debilitada a la condición "δ_j poco negativo", expresión que nos permite detener el procedimiento si en un tiempo razonable se ha obtenido un valor δ_j no negativo, o si es negativo pero no inferior a un número negativo preestablecido a priori por el decisor- porque sólo el decisor puede saber cuando una solución no exacta es aceptable.

Naturalmente si el proceso se detiene cuando δ_j es positivo, se obtendrá una solución exacta y se dirá que se cumple la condición de optimalidad en forma total, mientras que si δ_j tiene un valor negativo, se obtendrá una solución aceptable, y entonces el grado de cumplimiento de la condición de parada no será total.

La expresión correcta de esta situación se obtiene cuando se define mediante un conjunto difuso el conjunto de números que satisfacen la condición de parada debilitada. La notación que se suele usar para representar dicho conjunto difuso es la siguiente:

$$\delta_j \geq_f 0 \tag{2.10}$$

que definida por medio de su función de pertenencia:

$$\mu_j(\delta_j) = \begin{cases} 0 & \delta_j < -t_j \\ h_j(\delta_j) & -t_j \leq \delta_j \leq 0 \\ 1 & \delta_j > 0 \end{cases} \tag{2.11}$$

donde $h_j(.)$ es una función continua creciente tal que $h_j(-t_j) = 0$ y $h_j(0) = 1$ y $t_j > 0$ es el margen de tolerancia del incumplimiento de la condición de parada óptima. La gráfica de la función de pertenencia se muestra en la Figura 2.3.

Tanto la función h_j como t_j podrían ser dadas por el decisor como en [81], sin embargo sería de mejor ayuda para el decisor que el mismo criterio de parada difuso incluyera el mecanismo de construcción de la función de pertenencia así como el valor de t_j. Conocida esta referencia, el decisor podrá fijar el grado $\alpha \in (0,1]$ de incumplimiento de la condición de

parada exacta para que una solución no óptima pueda ser aceptable; así, la condición de parada (b) se modifica en el sentido de:

$$\mu_j(\delta_j) \geq \alpha, \quad \alpha\in(0,1], \quad j=1,...,n$$

equivalentemente

$$\delta_j \geq \mu_j^{-1}(\alpha), \quad \alpha\in(0,1], \quad j=1,...,n \tag{2.12}$$

Teniendo en cuenta que la condición $\mu_j^{-1}(\alpha)>0$ no tiene mayor importancia adicional en la solución sino es equivalente a $\mu_j^{-1}(\alpha) \geq 0$, la condición (2.12) se puede escribir en la siguiente forma:

$$\delta_j \geq h_j^{-1}(\alpha), \quad \alpha\in(0,1], \quad j=1,...,n. \tag{2.13}$$

Por tanto el criterio de parada (b), del algoritmo simplex puede ser flexibilizado mediante el siguiente

Criterio de Parada difuso:

b') Si $\delta_j \geq h_j^{-1}(\alpha)$, $\alpha\in(0,1]$, para todo $j \notin \Im$ (j=1,2,...,n), detener el proceso y la solución actual es aceptable.

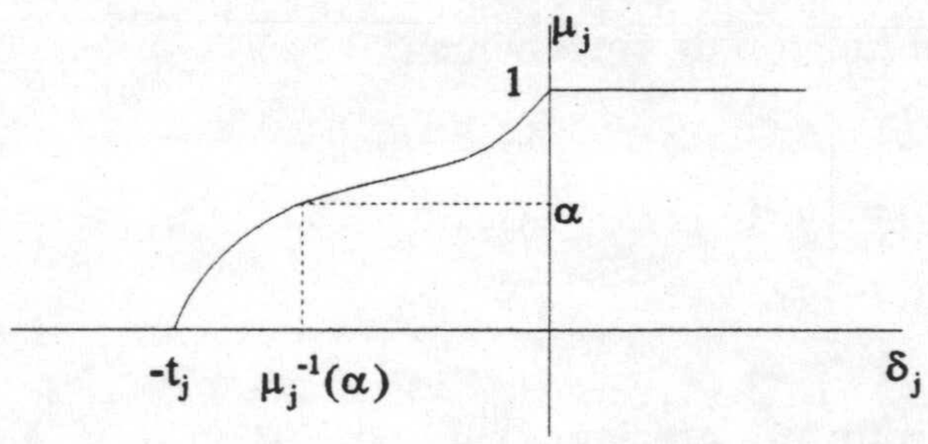


Figura 2.3. Función de pertenencia del criterio de parada difusa del algoritmo simplex

No obstante surgen algunos problemas al aplicar el algoritmo simplex, como por ejemplo: δ_j puede tener el mismo valor δ en cada iteración o a partir de una determinada iteración. Un ejemplo típico del primer caso es el problema de Klee-Minty [103]. En estas situaciones el criterio difuso no tendría sentido si $\delta < h_j^{-1}(\alpha)$, ya que la iteración continuaría

hasta alcanzar el valor óptimo, con lo cual la modificación propuesta no tendría los efectos deseados.

Este último inconveniente se puede evitar de la siguiente manera:

1) Tengamos en cuenta que con el criterio difuso pretendemos ahorrarnos el mayor número de iteraciones, deteniéndose las iteraciones cuando se haya alcanzado una solución aceptable. Quiere decirse que, a mayor número de iteraciones tendremos una mejor solución aceptable; por tanto, la convergencia de δ_j a la condición de parada exacta 0, en cierto modo depende también del número de iteraciones, por lo que utilizaremos δ_j/u como función medida para el criterio de parada difuso, donde u es el número de la iteración actual correspondiente.
2) Por otro lado, se sabe que para problemas de mayores dimensiones el número de iteraciones en el algoritmo simplex crece proporcionalmente al número de restricciones m, y es poco sensible al número de variables de decisión [155]. Por lo tanto, para compensar la división de δ_j entre u de tal manera que el criterio de parada no se modifique sustancialmente es necesario también dividir $h_j^{-1}(\alpha)$ entre m.

De esta manera, tenemos el siguiente nuevo criterio de parada difuso, que llamamos ***criterio de parada difuso balanceado***:

b") Si $\delta_j/u \geq h_j^{-1}(\alpha)/m$, $\alpha \in (0,1]$, para todo $j \notin \Im$ (j=1,2,...,n), detener el proceso y la solución actual es aceptable, donde u es la iteración actual, m número de restricciones y n el número de variables en el problema original.

Si la función h_j que define a la función de pertenencia es lineal, la expresión (2.13) se transforma en:

$$\delta_j /u \geq -t_j(1-\alpha)/m, \quad \alpha \in (0,1], \quad j=1,...,n. \tag{2.14}$$

Nivel de Tolerancia y función de Pertenencia:

En lo que sigue nos referiremos al nivel de tolerancia t_j y a la función de pertenencia $h_j(.)$. j=1,...,n.

Partiendo que la condición de parada para obtener la solución óptima es igual para todo j=1,...,n, asumimos que también t_j y el grado de cumplimiento α de la condición de parada deben ser iguales para todos los j=1,...,n. Por otro lado, la solución inicial es factible pero aceptable con un grado de aceptación ínfima, es decir, que ningún decisor puede admitirla como aceptable, a menos que sea óptima. Evidentemente, la solución aceptable se obtendrá entre la primera iteración y la iteración correspondiente a la solución óptima; en consecuencia, el valor común de los t_j debe estar dado por el menor de los δ_j^0 obtenidos en la primera iteración, es decir,

$$t_j=r_0, \quad \forall\, j=1,...,n, \quad r_0=\min\{\delta_j^0 / j=1,...,n\} \tag{2.15}$$

Con esta opción se está considerando que la solución inicial, de no ser óptima, es una solución aceptable con el grado de cumplimiento cero.

Para definir la función de pertenencia, primero tenemos que decidir si usamos una función $h_j(.)$ lineal o no lineal; si optamos por una función lineal, (2.14) será la condición de parada difusa. Si preferimos que $h_j(.)$ sea no lineal, tendríamos que definirla. Una vez que se tiene definida la función $h_j(.)$, como siguiente paso, el decisor tendrá que fijar el grado mínimo α de aceptación de una solución no óptima.

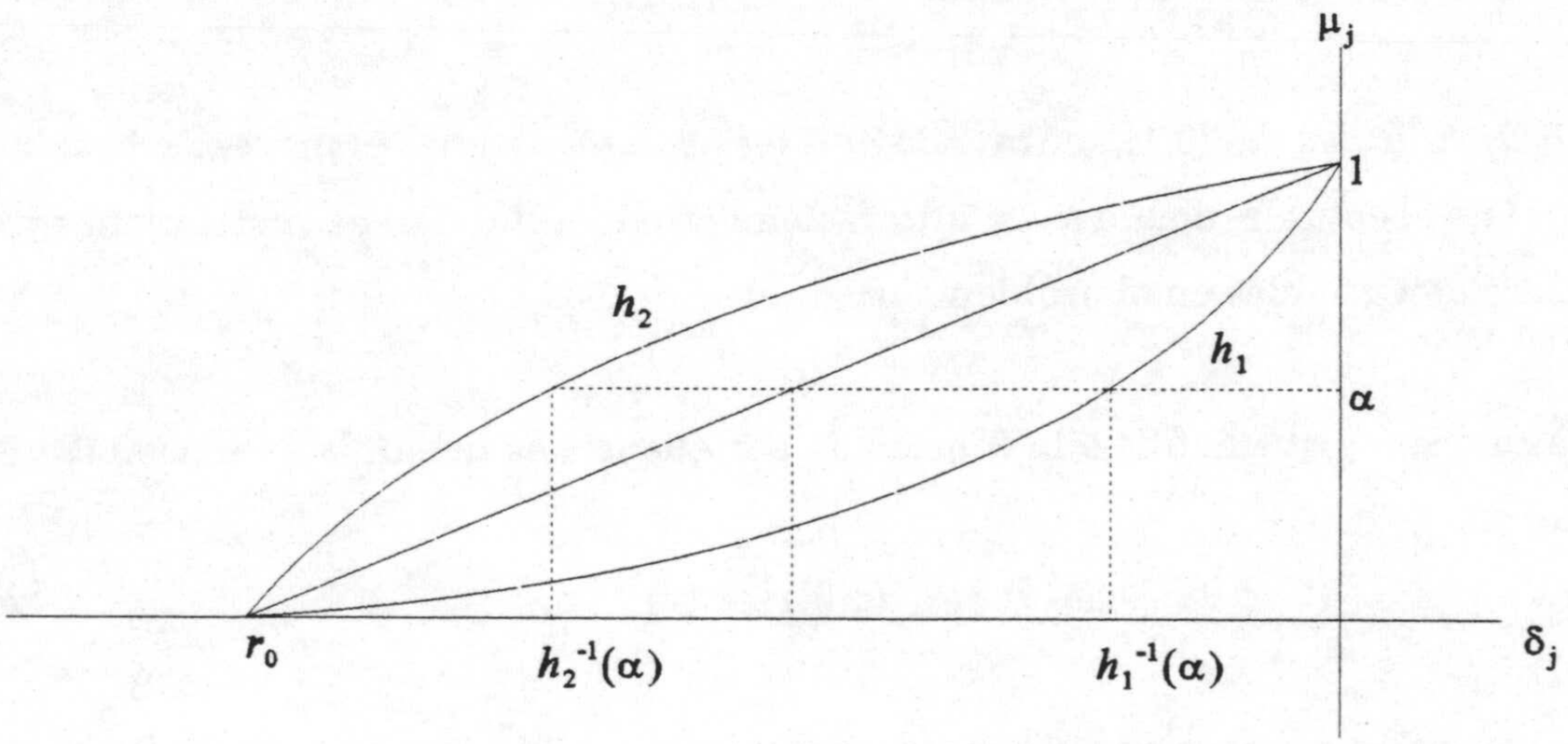


Figura 2.4. Funciones de pertenencia h_1 (convexa), h_2 (cóncava) y lineal en el criterio de parada difusa del algoritmo simplex.

Es importante destacar que dado α , que expresa el grado de cumplimiento de la condición de parada, una función no lineal convexa en $[r_0,0]$ proporcionará mejor solución que una función lineal ó no lineal cóncava, para un mismo grado de pertenencia. Pues si h_1 es convexa y h_2 es cóncava en $[r_0,0]$, entonces $h_1^{-1}(\alpha) > h_2^{-1}(\alpha)$, tal como se muestra en la Figura 2.4; esto último implica que para alcanzar la condición de parada difusa (b") se necesitará mayor número de iteraciones cuando la función sea convexa que cuando la función sea cóncava.

Dos ejemplos de funciones que se pueden utilizar para definir las funciones de pertenencia, entre otros muchos, son los siguientes:

$$h_1(t) = \frac{\exp(-r_0+t)-1}{\exp(-r_0)-1} \tag{2.16}$$

una función convexa y

$$h_2(t) = \sqrt{\frac{r_0-t}{r_0}} \tag{2.17}$$

una función cóncava, donde r_0 es la tolerancia mínima de incumplimiento de la condición de parada.

Ejemplo:

Como ilustración resolvemos el siguiente problema:

$$\begin{aligned} \max \quad z &= 4x_1+6x_2+8x_3+5x_4 \\ s.a. \qquad x_1+3x_2+2x_3+4x_4 &\le 20 \\ 2x_1+3x_2+6x_3+4x_4 &\le 25 \\ x_1+x_2 &\le 5 \\ x_1+2x_2 &\le 8 \\ 4_3+3x_4 &\le 12 \\ x_1, x_2, x_3, x_4 &\ge 0. \end{aligned} \tag{2.18}$$

Aplicando el algoritmo simplex, se obtiene la solución óptima siguiente:

$x = (x_1, x_2, x_3, x_4) = (2,3,2,0)$, y la utilidad total es $z = 42$.

Para ejemplificar el criterio de parada difuso en el problema anterior, utilizaremos tres funciones: una función convexa y una función cóncava, según (2.16) y (2.17) respectivamente, y una función lineal. A partir de la solución inicial se obtiene r_0=-8 que utilizaremos como nivel de tolerancia mínima del incumplimiento de la condición de parada. Las Figuras 2.5, 2.6 y 2.7 muestran las funciones de pertenencia usadas.

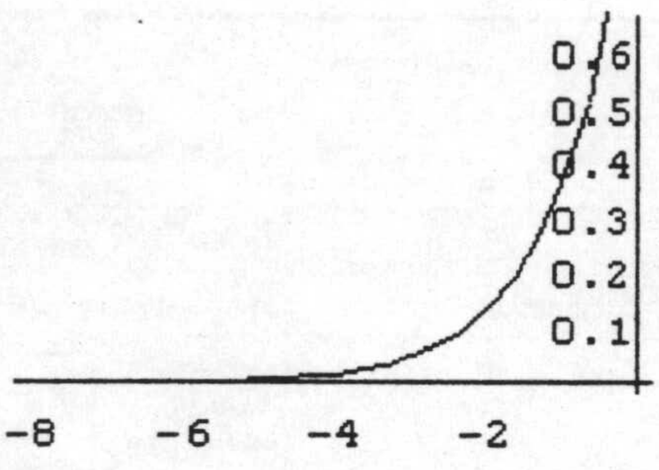


Figura. 2.5. Función convexa

$h(t)=(e^{8+x}-1)/(e^8-1)$

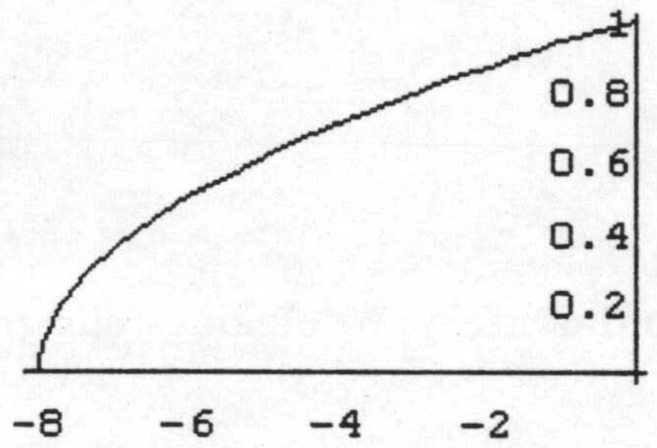


Figura 2.6. Función cóncava

$h(t)=(1+x/8)^{1/2}$

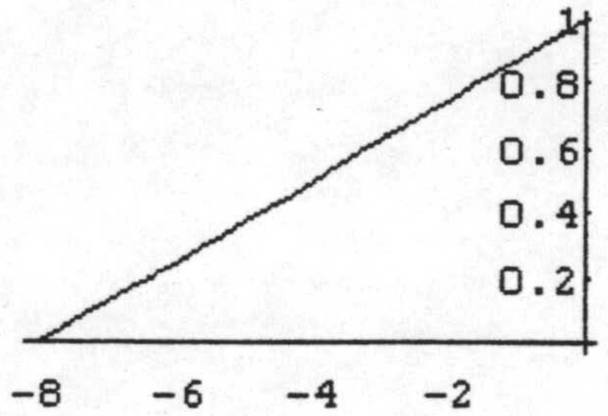


Figura 2.7. Función lineal

$h(t)=1+x/8$

En los resultados de los cálculos que se muestran en la Tabla 2.1. se observa que, cuando se utiliza una función $h(t)$ convexa para definir la función de pertenencia de la regla de parada difusa, para un mismo grado de aproximación α, se obtiene una solución mucho más cercana a la óptima que cuando se utiliza una función lineal o una función cóncava. Particularmente, para α=0.5, usando la función convexa, se ha obtenido la solución z=41.33 muy cercana a la solución óptima, mientras que usando la función lineal y usando la función cóncava se obtiene z=38, que es una solución no tan cercana a la óptima por tanto con menor grado de aceptación que el anterior. Cuando se aumenta el grado de aproximación a α=0.8, en caso de usar la función convexa se ha obtenido la solución exacta, mientras que en el caso lineal se mejora la solución no exacta pero no se ha mejorado en el caso de usar una función cóncava.

Criterio de parada difuso		**Solución**	
Grado de aceptación α	**Condición de parada $\delta/u \geq h^{-1}(\alpha)/m$**	**Punto (vértice) (x_1,x_2, x_3, x_4)**	**Utilidad z**
0.5	$\delta/u \leq$-0.138 (convexa)	(0,4,13/6,0)	41.33
	$\delta/u \leq$-0.800 (lineal)	(0,7/3,3,0)	38
	$\delta/u \leq$-1.200 (cóncava)	(0,7/3,3,0)	38
0.8	$\delta/u \leq$-0.045 (Convexa)	**(2,3,2,0)**	**42**
	$\delta/u \leq$-0.320 (Lineal)	(0,4,13/6,0)	41.33
	$\delta/u \leq$-0.578 (cóncava)	(0,7/3,3,0)	38

Tabla 2.1. Soluciones obtenidas mediante criterios de parada difusos

2.3 CRITERIOS DE PARADA DIFUSOS EN EL ALGORITMO DE KARMARKAR

2.3.1 Introducción

Desde que Khachiyan [102] en 1979 propuso el algoritmo elipsoidal para resolver un problema de PL, se han buscado otros métodos, que no sólo ignoren la estructura combinatoria sino que mejoren el tiempo de ejecución polinomial del algoritmo elipsoidal.

Así, en 1984 Karmarkar [99] propone un nuevo algoritmo de tiempo de ejecución polinomial mucho más rápido que el algoritmo elipsoidal, por tanto un mejor competidor del anterior algoritmo simplex, pero que al igual que el algoritmo elipsoidal ignora la característica combinatoria. El algoritmo de Karmarkar, conocido también como el método de transformación proyectiva, genera una sucesión de puntos interiores de la región factible que con el aumento de las iteraciones se aproxima a los extremos hasta alcanzar el vértice óptimo. Por presentar esta característica, el algoritmo de Karmarkar es un método de punto interior.

Debido a que este algoritmo no es tan popular como el algoritmo simplex, hacemos una revisión con mayor detalle incidiendo mayormente en la parte intuitiva y en los mismos pasos del algoritmo en sí.

2.3.2 Ideas básicas

Como sabemos un problema de PL consiste en determinar x de tal que:

$$\begin{aligned} &\min cx \\ &x \in D \subset \mathbb{R}^n, \end{aligned} \tag{2.19}$$

donde D es la intersección de varios semiespacios de la forma $Ax \geq 0$.

Si la región D es un sólido esférico de centro v y radio r, para resolver el problema (2.19), es decir, para hallar x_0 en D donde $f(x) = cx$ alcanza su mínimo valor, se utiliza la propiedad del gradiente ∇f de que $-\nabla f$ tiene la dirección de decrecimiento más rápido de la función f; por lo tanto el punto mínimo es

$$v+\frac{r(-c)}{\|c\|} \tag{2.20}$$

que está en la esfera ∂D, porque $\nabla f=c$.

Ahora supongamos que la región D es la intersección de un espacio afín y un sólido esférico con centro en el espacio afín, es decir, $D=\{x\in\mathbb{R}^n / Ax=b\}\cap S(v,r)$, $(Av=b)$, es un sólido esférico de dimensión menor que S. En este caso el punto (2.20) no se encuentra en D.

Para seguir la misma idea que en el caso anterior, ahora podemos considerar la proyección c_p de c sobre el hiperplano $\{x\in\mathbb{R}^n / Ax=0\}$. Entonces $-c_p$ es la dirección del decrecimiento más rápido de la función objetivo en D, y el punto mínimo es:

$$v+\frac{r(-c_p)}{\|c_p\|} \tag{2.21}$$

Evidentemente, se observa que si D es un sólido esférico es muy fácil resolver el problema (2.19). Sin embargo los problemas de PL no tienen estas características. El objetivo es aprovechar la sencillez de la obtención de (2.21) para resolver el problema de PL (2.19), y se consigue de la siguiente manera [98]:

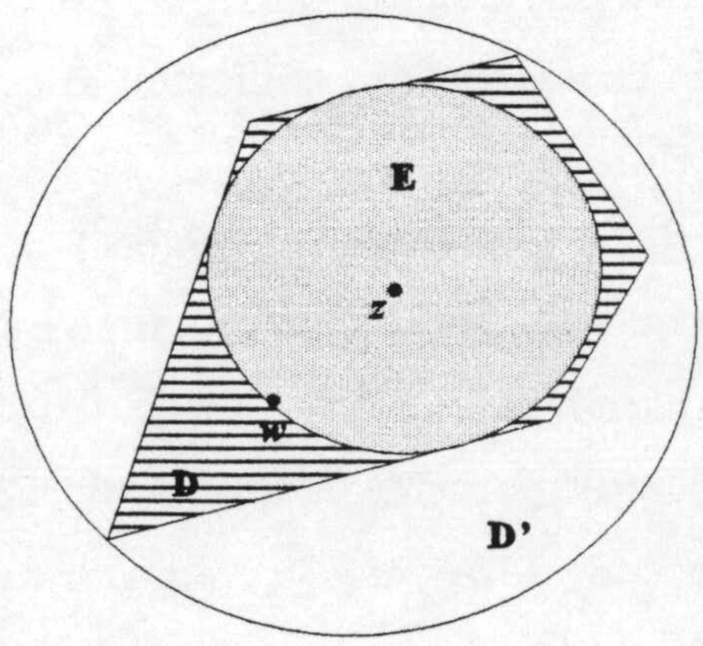


Figura 2.8. Redondez de la región D, respecto a z.

Supongamos que se tiene un punto factible z. Hay que construir un sólido esférico (de mayor volumen posible) contenido en la región factible D (Fig. 2.8), y luego determinar el punto mínimo w como en (2.21). Utilizando este nuevo punto w , se repite el proceso. Así, después de un número de repeticiones se obtendrá una solución aproximada del problema. La solución obtenida será tanto mejor aproximación cuanto la región factible más se parezca a un sólido esférico.

Para medir el grado en que se parece la región factible a un sólido esférico respecto a un punto, se puede utilizar la relación entre el radio de la esfera contenida (de mayor volumen), y el radio de la esfera (de menor volumen) que contiene a la región factible. Esta relación se denomina medida de la *redondez.*

Si m es el valor mínimo buscado en (2.19), si w es el punto mínimo en el sólido esférico E=S(z,r) (de mayor radio posible) contenido en D , y si E'=S(z,kr) es el menor sólido esférico que contiene a D (E' es k veces mas grande que E) [98], se tiene que:

$$cw - m \leq (1 - \frac{1}{k})(cz - m) \quad . \tag{2.22}$$

Teniendo en cuenta que $k \geq 1$, de (2.22) se deduce que cuanto mayor sea k, la diferencia entre el mínimo valor de la función objetivo del problema (2.19) con el mínimo en E (en w) es más cercana a la diferencia entre el mínimo y el valor en z, mientras que si k es cercano a 1, w es muy cercano al punto mínimo en D.

En consecuencia el punto w sería una mejor aproximación al punto mínimo en D si k es muy cercano a 1, pero esto sólo se cumple si:

- D es muy semejante a un círculo (polígono regular de n lados).
- z es el centro de D (en el caso de que sea un polígono regular z sería el centro de las circunferencias inscrita E y circunscrita E')

En este caso se diría que D es muy parecido a un círculo o que es un polítopo *redondeado.*

Karmarkar, aprovechando estos hechos, propone un método de solución de un problema de PL: toma una región factible redondeada que le permita elegir como punto factible inicial el centro, y utilizando una transformación adecuada en cada una de las

iteraciones, mantiene esta característica. La función objetivo debe ser no negativa y tal que el mínimo valor sea cero. El problema con estas características se denomina *forma estándar de Karmarkar*. Resumimos a continuación el algoritmo de Karmarkar para la forma estándar y luego su aplicación al caso general.

2.3.3 Algoritmo de Karmarkar

El algoritmo de Karmarkar no resuelve directamente el PLE dado en (2.6), sino tiene su propia forma estándar que veremos en 2.3.3.1, luego en la sub-sección 2.3.3.2 revisaremos el procedimiento y finalmente en la sub-sección.2.3.3.3 la aplicación a todos los casos generales de la problema de PL.

2.3.3.1 *Forma estándar de Karmarkar de programas lineales*

Sea

$$\Delta = \left\{ x \in R^n \,/\, x \geq 0, \sum_{j=1}^{n} x_j = 1 \right\} \tag{2.23}$$

el *simplex estándar (n-1)-dimensional* con *centro*

$$a_0 = \frac{1}{n} e \quad , e=(1,1,...,1). \tag{2.24}$$

Sea A una matriz *mxn* de números enteros, *c* un vector *n*-dimensional de números enteros y sea

$$\Omega = \left\{ x \in R^n \,/\, Ax = 0 \right\} .$$

Entonces el *programa lineal estándar de Karmarkar* es:

$$\begin{aligned} \min \quad & z = cx \\ \text{s.a.} \quad & x \in \Pi = \Omega \cap \Delta \\ & x, z \geq 0 \end{aligned} \tag{2.25}$$

o equivalentemente :

$$\begin{aligned} \min \quad & z = cx \\ \text{s.a.} \quad & Ax = 0 \\ & x_1+x_2+\ldots+x_n = 1 \\ & x,z \geq 0 \end{aligned} \qquad (2.25')$$

2.3.3.2 *Procedimientos de solución*

El algoritmo de Karmarkar tiene dos características fundamentales:

a) En cada iteración halla el punto mínimo en una bola contenida en Π y que exprese la mejor redondez.
b) En cada iteración el problema siempre es de forma estándar, es decir, semejante a Π.

Karmarkar utiliza el simplex Δ por sus características siguientes:

- Es redondeado sobre el punto a_0 definido en (2.24).
- $B(a_0,r)$ es la mayor bola contenida en Δ. y
- $B(a_0,R)$ es la menor bola que contiene a Δ.

donde

$$r = \frac{1}{\sqrt{n(n-1)}} \qquad R = \sqrt{\frac{n-1}{n}} \qquad (2.26)$$

Por ello, elige a_0 como punto inicial; a partir de este punto y siguiendo la dirección del decrecimiento más rápido halla el punto mínimo a sobre una esfera de centro en este punto. Si el punto mínimo así obtenido no es el punto mínimo global (con costo cero), repite el procedimiento anterior utilizando el punto a. Es evidente que el punto a ya no es el centro del simplex Δ, y por tanto no se puede repetir exactamente el mismo procedimiento. Para superar este inconveniente, Karmarkar define una transformación proyectiva particular T_a de tal manera que al punto a le hace corresponder con el punto a_0. Por suerte la transformación también cumple con la condición $T_a(\Delta) = \Delta$, y está definida de la siguiente manera:

$$T_a(x)=x', \quad \mathrm{x}'=(\mathrm{x}'_1,\ldots,x'_n), \quad x'_j=\frac{x_j/a_j}{\sum_{j=1}^{n} x_j/a_j} \qquad (2.27)$$

o en forma compacta (matricial)

$$T_a(x) = \frac{Cx}{e^T Cx} = \frac{D^{-1}x}{e^T D^{-1}x} \tag{2.28}$$

donde:

$$D = C^{-1} = \begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & a_n \end{bmatrix}, \qquad e^T = (1,1,\ldots,1). \tag{2.29}$$

Debido a que $a>0$, D es una matriz no singular y T_a es inversible. Además,

- $Ax=0$ si y sólo sí $ADx'=0$, por lo tanto, si $\Omega=\{x/Ax=0\}$ y $\Omega'=\{x/ADx'=0\}$. Entonces $T_a(\Omega)=\Omega'$ y si $a\in\Omega$, $a_0\in\Omega'$.
- $T_a(\Delta\cap\Omega)=\Delta\cap\Omega'$. Haciendo $\Pi'=\Delta\cap\Omega'$, se tiene $T_a(\Pi)=\Pi'$.
- Π' es redondeado pues fácilmente se deduce que $B'(a_0,r)\subseteq\Pi\subseteq B'(a_0,R)$, donde $B'(a_0,r)=B(a_0,r)\cap\Omega'$ y $B'(a_0,R)=B(a_0,R)\cap\Omega'$.

La función objetivo lineal original se transforma en la siguiente:

$$\frac{\sum_{j=1}^{n}(c_j a_j)x'_j}{\sum_{l=1}^{n} a_l x'_l} \tag{2.30}$$

que es una función *no lineal*. Si el costo mínimo es cero, y teniendo en cuenta que la minimización de (2.30) (una función racional) es sobre un polítopo, es suficiente minimizar el numerador, que es lineal. Sin embargo, en el caso de que el mínimo no fuera cero no sería correcto este procedimiento.

Para superar este inconveniente, Karmarkar, utiliza la *función potencial*

$$f(x) = \ln\left[\frac{(cx)^n}{x_1 \cdots x_n}\right] \tag{2.31}$$

donde cx es la función objetivo y $x\in\Pi$, $x>0$.

Esta función permite controlar la convergencia al punto mínimo que puede ser cero o diferente de cero. Si el costo mínimo en Π es cero (0), Karmarkar demuestra que si en cada iteración se obtiene un nuevo punto b tal que $f(b)\leq f(a)-\delta$, (δ constante), se garantiza

la *convergencia* en un tiempo polinomial y el valor de convergencia del costo es cero (0). De este resultado se deduce que si el costo mínimo en b no es cero y $f(b) \geq f(a)-\delta$, el costo mínimo en Π no es cero. Estos resultados son debidos a la propiedad de invariante de f con respecto a la transformación T_a.

Para garantizar que este pequeño cambio no altera la idea del cálculo de b dentro de una bola contenida en Π', Karmarkar elige la esfera $S(a_0, \alpha/n)$ con $(1/n)<r$ (r dado en (2.26)), así el nuevo problema de minimización es hallar el punto mínimo b' sobre $S(a_0, \alpha/n) \cap \Pi'$.

Pasos del Algoritmo de Karmarkar:

Sea

$$\delta = \alpha - \frac{\alpha^2}{1-\alpha} > 0, \qquad 0 < \alpha < 1/2 \tag{2.32}$$

y sea $L=(m+1)n+\lceil \log_2 |P| \rceil + n\lceil \log_2 n \rceil$, donde P es el producto de todos los coeficientes no nulos de la función objetivo y de las $m+1$ restricciones, y $\lceil . \rceil$ es el número entero inmediato superior. Al conjunto de operaciones para obtener un nuevo punto b a partir de un punto a conocido en cada iteración se le nota por Φ, es decir $b=\Phi(a)$.

Etapa Inicial: Sea $x^{(0)}=a_0=e/n$ y sea $K=\lceil 2nL/\delta \rceil$ (el número de iteraciones).

Criterios de parada: si $k<K$ y $cx^{(k)}=0$, $v=x^{(k)}$.

Si $k<K$ y $f(x^{(k)})>f(x^{(k-1)})-\delta$, el costo óptimo es positivo, v es un punto de costo no mayor que $cx^{(K)}$.

Etapa iterativa: Para k=1 a K, Calcular: $x^{(k)} = \Phi(x^{(k-1)})$.

La solución óptima que se obtiene es el punto v. Si el mínimo de la función objetivo es no nulo, este punto v es aquel que tiene el menor costo del total de los K (K~12nL) puntos revisados durante las iteraciones.

Cálculo de $b = \Phi(a)$:

Dado $a \in \Pi$, $a>0$ y $ca>0$, con c vector de costos hallar $b=\Phi(a)$ que está en Π y $b>0$:

1. Sea $D = \begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & a_n \end{bmatrix}$, para todo x, $T_a(x) = \frac{D^{-1}x}{e^T D^{-1}x}$ y $T_a^{-1}(x') = \frac{Dx'}{e^T Dx'}$.

2. Sea $c' = Dc$

3. Sea $B = \begin{bmatrix} AD \\ e^T \end{bmatrix}$, con A matriz de los coeficientes tecnológicos.

($\Pi' = \{ x' / Bx' = e' \}$, e' tiene todos sus componentes nulos excepto el último que es 1.)

4. Sea $c_P = c' - B^T(BB^T)^{-1} Bc'$ (la proyección de c' sobre el espacio nulo de B)

5. Si $c_P = 0$, sea $\hat{c}_P = 0$. En otro caso $\hat{c}_P = \frac{c_P}{\|c_P\|}$ un vector unitario con la misma dirección de c_P .

6. Luego el punto mínimo es: $b' = a_0 - \frac{\alpha}{n}\hat{c}_P$.

7. Retornar: $b = T_a^{-1}(b') = \frac{Db'}{e^T Db'}$

8. FIN.

Observación: La complejidad de este algoritmo es del orden $O(n^4L)$, y un máximo de 12nL iteraciones aproximadamente.

2.3.3.3 Aplicación en los problemas de PL

Recordemos que un problema general de PL puede ser de maximización o minimización. Las restricciones pueden ser desigualdades de >, <, ≤ o ≥. Las variables no necesariamente pueden ser positivas. Todos estos casos, ya sea cambiando de signo, ya sea

introduciendo variables de holgura y/o variables positivas, se pueden transformar en un problema de minimización y con las restricciones de igualdad como sigue:

$$\begin{aligned} \min\ & z = cx \\ \text{s.a. } & Ax = b \\ & x \geq 0 \end{aligned} \tag{2.33}$$

En general la función objetivo no es no negativa, y el mínimo no siempre es cero. En algunos casos a partir de las condiciones de no negatividad se puede determinar que la función objetivo es no negativa aunque el mínimo no necesariamente sea cero.

Si se desconoce que la función objetivo es no negativa, el problema (2.33) se debe transformar en un sistema de ecuaciones, considerando las restricciones del problema primal, del problema dual y la igualdad de las funciones objetivo en el punto óptimo, luego realizar la transformación siguiente.

Si el programa es factible, toda solución básica factible tiene cada uno de sus componentes no mayor que 2^L, donde L es la dimensión de $Ax = b$. Entonces , si A es una matriz de dimensión mxl, para todo punto factible x se cumple:

$$x_1 + x_2 + \ldots + x_l \leq M, \qquad M = l2^L.$$

e introduciendo otra variable de holgura se tiene que

$$x_1 + x_2 + \ldots + x_l + x_{l+1} = M$$

Luego la región factible queda descrita por:

$$\begin{aligned} Ax &= b \\ x_1 + x_2 + \ldots + x_l + x_{l+1} &= M \\ x &\geq 0, \end{aligned}$$

donde $x = (x_1, x_2, \ldots, x_l, x_{l+1})$, y A es la matriz anterior aumentada en una columna de ceros.

La sustitución $x' = x/M$ da lugar a las siguientes expresiones

$$\begin{aligned} (MA)x' &= b \\ x'_1 + x'_2 + \ldots + x'_l + x'_{l+1} &= 1 \\ x' &\geq 0, \end{aligned}$$

Finalmente, restando en cada fila $b_i(x_1+x_2+\ldots+x_l+x_{l+1}) = b_i(1)$ miembro a miembro se obtiene:

$$\begin{aligned} Cx &= 0 \\ x_1 + x_2 + \ldots + x_1 + x_{1+1} &= 1 \\ x &\geq 0, \end{aligned} \tag{2.34}$$

donde en lugar de x' se ha escrito x y

$$C = \begin{bmatrix} Ma_{11} - b_1 & Ma_{12} - b_1 & \cdots & Ma_{1l} - b_1 & -b_1 \\ Ma_{21} - b_2 & Ma_{22} - b_2 & \cdots & Ma_{2l} - b_2 & -b_2 \\ \vdots & & & & \\ Ma_{m1} - b_m & Ma_{m2} - b_m & \cdots & Ma_{ml} - b_m & -b_m \end{bmatrix}.$$

El punto a_0 no es una solución factible del sistema (2.34). Para que lo sea, se agrega una nueva variable λ en las ecuaciones del sistema dado en (2.34), y se transforma en:

$$\begin{aligned} Cx - \lambda(Ce) &= 0 \\ x_1 + x_2 + \ldots + x_1 + x_{1+1} + \lambda &= 1 \\ x, \lambda &\geq 0, \end{aligned} \tag{2.35}$$

Luego el problema (2.33) consiste en **minimizar z= cx+λ** sujeto a las restricciones de (2.35) si la función objetivo inicial es no negativa, y **minimizar z = λ** si se desconoce la no negatividad de la función objetivo inicial, donde c es el costo inicial multiplicado por M.

Ejemplo: Ilustramos con un ejemplo la forma como se construye una forma estándar de Karmarkar a partir de un problema arbitrario:

Consideremos el modelo:

$$\begin{aligned} \min\ & x+y \\ s.a.\ & x+y \geq 1 \\ & x-y \leq 1 \\ & x,y \geq 0 \end{aligned}$$

escribiendo las restricciones como igualdad se obtiene:

$$\begin{aligned} \min\ & x+y \\ s.a.\ & x+y-z = 1 \\ & x-y+u = 1 \\ & x,y,z,u \geq 0 \end{aligned}$$

Éste último tiene m=2 ecuaciones y n=4 variables, por lo tanto su tamaño es L=20, además la función objetivo es no negativa, en consecuencia no es necesario transformar el problema en un sistema de ecuaciones, sino directamente transformamos en la forma estándar de Karmarkar.

Puesto que L = 20, se tienen $x+y+z+u \leq 4(2^{20})$ e introduciendo una variable de holgura se tiene el nuevo problema:

$$\min\ x+y$$

$$\begin{aligned} s.a. \qquad x+y-z &= 1 \\ x-y+u &= 1 \\ x+y+z+u+v &= 2^{22} \\ x,y,z,u &\geq 0 \end{aligned}$$

Sustituyendo cada variable por 2^{22} veces otra variable y luego procediendo a anular los segundos miembros, y finalmente introduciendo λ (teniendo en cuenta todo el procedimiento explicado arriba) se tiene

$$\min\ 2^{22}x_1+2^{22}x_2+\lambda$$

$$\begin{aligned} s.a. \qquad (-1+2^{22})x_1+(-1+2^{22})x_2+(-1-2^{22})x_3-x_4-x_5+(5-2^{22})\lambda &= 0 \\ (-1+2^{22})x_1+(-1-2^{22})x_2-x_3+(-1+2^{22})x_4-x_5+(5-2^{22})\lambda &= 0 \\ x_1+x_2+x_3+x_4+x_5+\lambda &= 1 \\ x_1,x_2,x_3,x_4,x_5,\lambda &\geq 0 \end{aligned} \qquad (2.36)$$

que constituye el problema en la forma estándar de Karmarkar, asociado al problema propuesto.

El problema (2.36) es de dimensión L=262, por lo tanto el número máximo de iteraciones es de 18864 como mínimo, y el tiempo de ejecución del O(339552).

2.2.4 El criterio de parada difuso en el algoritmo de Karmarkar

Se observa que en el ejemplo anterior, siendo el problema el más simple posible, el número de iteraciones es muy elevado. En problemas con mayor número de variables, y con

coeficientes diferentes de 1, el tamaño se incrementa y consecuentemente el número de iteraciones o más aún, el número de cálculos, es extremadamente elevado, por ejemplo en el problema AFIRO [3] de n=32 variables y L=6352 requiere 2439168 iteraciones ó 6660554752 operaciones de cálculo, en el que una máquina capaz de efectuar un billón de operaciones por segundo lo puede concluir en 6 segundos, mientras que, la misma máquina para resolver el problema FIT2P [3] con n=13525 y L=3518352 necesitaría 3733204 años.

Como se ve, en problemas que aún no son el "peor caso", el número de iteraciones es muy elevado, de modo que en tiempo razonable no es posible obtener la solución. En estos casos se hace necesario detener el proceso de solución cuando se haya obtenido una solución razonablemente aceptable para el decisor, aún no siendo la óptima [81].

Todo problema de PL de minimización que tenga región factible no vacía tiene un conjunto de soluciones. Cada uno de ellos tiene para el decisor un determinado grado de aceptabilidad. Evidentemente, un punto que no es factible es descartado, se puede decir que su aceptabilidad es nula, y por otro lado, un punto con costo muy alto tendrá un grado de aceptación muy baja (pero no nula), mientras que un punto con costo mínimo tendrá el más alto grado de aceptación. Desde este punto de vista resolver un problema de PL es hallar un punto con el mayor grado de aceptabilidad (óptimo).

Cuando se admite la posibilidad de que el decisor pueda aceptar como solución a aquél que tenga un alto grado de aceptabilidad aunque no tenga el mayor grado, se requiere una representación adecuada como es la representación mediante una función de pertenencia.

Si se conoce toda la región de factibilidad y el punto mínimo, la función de pertenencia sería fácil de definir pues el punto(s) mínimo(s) tendría el mayor grado de pertenencia y disminuiría a medida que el costo aumentara , siendo cero para aquellos puntos no factibles. Se observa que el valor de la función de pertenencia está en función del costo, por tanto mejor se define en términos del costo en lugar de definir en términos del punto factible.

En el caso de un problema nuevo se desconoce el punto mínimo e incluso la región de factibilidad, por tanto para definir la función de pertenencia se toma como referencia un costo c_m, que es el costo del punto inicial del algoritmo de Karmarkar, debido a que el punto

inicial de este algoritmo es factible. El costo mínimo será menor o igual a este costo c_m.

La función de pertenencia se define de la siguiente manera:

$$\mu(z) = \begin{cases} g(z) & \text{si} \quad 0 \le z \le c_m \\ 0 & \text{si} \quad z > c_m \end{cases} \tag{2.37}$$

donde la función g es continua monótona no creciente y $g(z) \in [0,1]\ \forall\ z$, como se ilustra en la Figura 2.9.

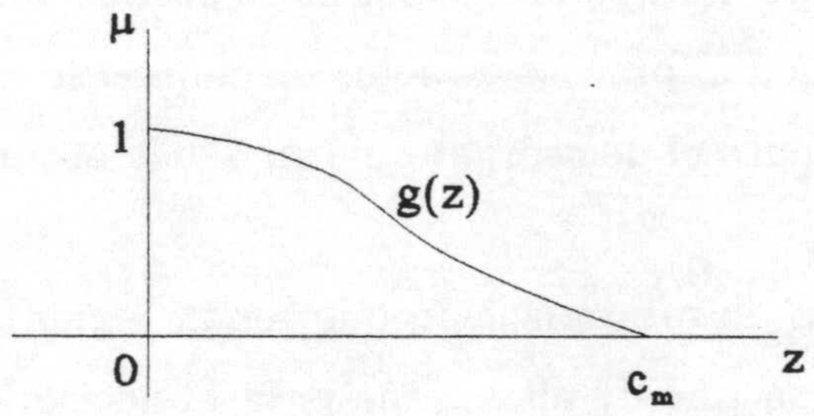


Figura 2.9. La función de pertenencia del costo

Si el decisor, frente al problema nuevo, decide admitir como solución el caso en que el costo tiene un grado de pertenencia no menor que $\alpha \in (0,1]$, es decir, si $\mu(cx) \ge \alpha$ o $g(cx) \ge \alpha$, equivalentemente $cx \le g^{-1}(\alpha)$, entonces la condición de parada del algoritmo de Karmarkar puede ser relajada a:

$$cx \le g^{-1}(\alpha) \tag{2.38}$$

donde g y α son dados por el decisor.

En un problema de minimización general, la solución óptima no necesariamente tiene que ser de costo cero (desde el punto de vista de las aplicaciones el costo siempre será no negativo) sino que puede ser de costo positivo e incluso negativo.

Al utilizar la condición de parada (2.38) se obtendrá una solución aproximada, y esta solución depende del valor de $g^{-1}(\alpha)$. Por lo tanto, éste tiene que ser adecuado de tal manera que el error cometido en la aproximación sea muy pequeño, pero determinado en un tiempo razonable. Para ello el decisor puede fijar el grado de pertenencia aceptable para el costo, y el que resuelve el problema debe definir la función de pertenencia.

Fijado el grado de pertenencia mínimo aceptable α, la calidad de la aproximación depende únicamente de la función . Al escoger esta función debe tenerse en cuenta la posibilidad de que el mínimo puede ser cero o mayor que cero.

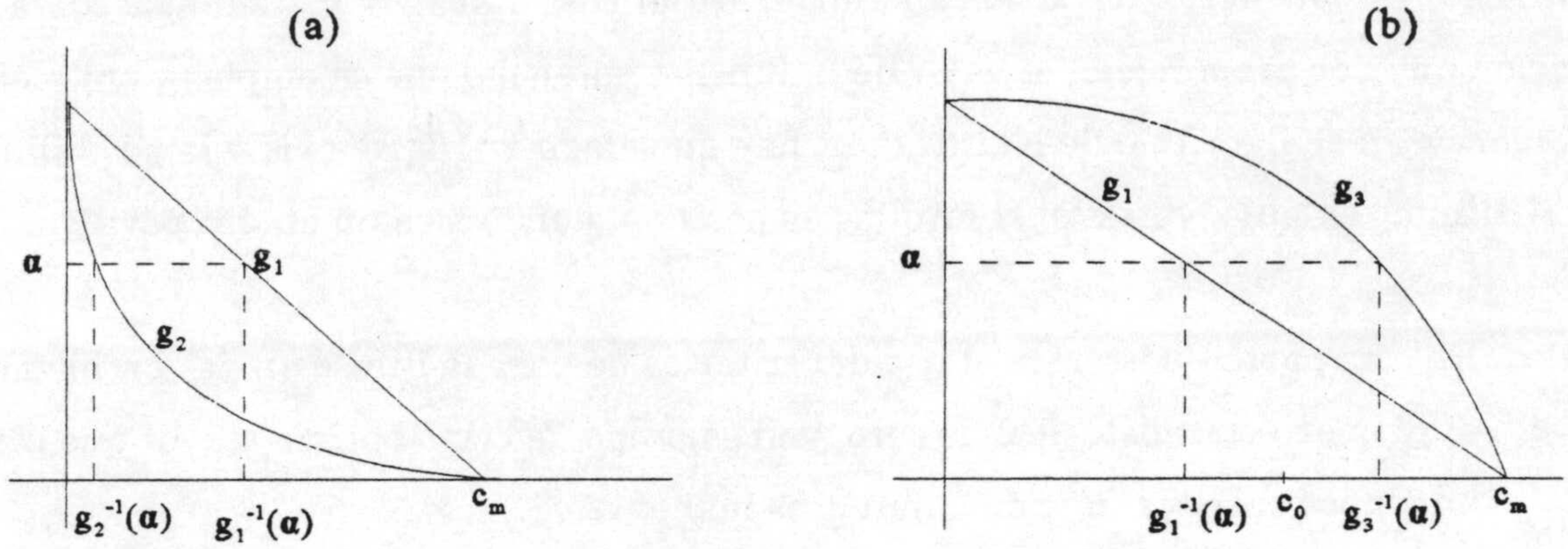


Figura. 2.10. Funciones de pertenencias: (a) para el caso de mínimo cero, y (b) caso mínimo positivo

Es costumbre utilizar la función lineal para definir la función de pertenencia, si en este caso también se opta por esta alternativa, existen varias posibilidades:

i) Si el costo óptimo (mínimo) es cero, la condición relajada (2.38) puede conducir a aceptar una solución aproximada con muchos errores, como se puede observar en la Figura. 2.10-(a).
ii) Si el costo óptimo es diferente de cero, existe la posibilidad de que el óptimo sea mayor que $g^{-1}(\alpha)$, como se ve en la Figura.2.10-(b). Esto conduciría a realizar todas las iteraciones para obtener la solución.

El primer caso se puede resolver utilizando una función de pertenencia convexa de la forma g_2, es decir, una asíntota con el eje de las ordenadas, mientras que en el segundo caso se puede superar utilizando una función cóncava como g_3.. Así se obtiene una mejor solución aproximada.

Siempre sigue latente la dificultad de optar por uno u otro caso, cuando a priori no se conoce si el mínimo es cero o positivo. En este caso la solución radica en realizar los ajustes

de la función objetivo durante el proceso de cálculo, utilizando la función siguiente:

$$g(x)= 1-(x/c_m)^r. \tag{2.39}$$

Durante las primeras iteraciones se utilizará el caso lineal r=1, y de acuerdo a la información que nos proporciona el valor de la función potencial que se emplea para medir la convergencia en el algoritmo de Karmarkar, hay que elegir r=1/k, si existe la posibilidad de que el mínimo es nulo y r=k, si el mínimo es positivo, donde k es un entero positivo.

Ejemplo: En esta parte ilustremos el uso del criterios de parada difuso en dos problemas, el primero con costo óptimo cero, y el segundo de costo óptimo mayor que cero, en ambos casos la función objetivo es no negativa.

1)
$$\begin{aligned} &\min\ 3x+y \\ s.a.\ \ &x-2y \le 0 \\ &2x-y \ge 0 \\ &x,y \ge 0 \end{aligned}$$

2)
$$\begin{aligned} &\min\ x+y \\ s.a.\ \ &x+y \ge 1 \\ &x-y \le 1 \\ &x,y \ge 0 \end{aligned}$$

Expresando, los problemas, en la forma estándar de Karmarkar, se obtiene respectivamente:

$$\min\ 3(2^{25})x_1+(2^{25})x_2$$

$$\begin{aligned} s.a.\ \ x_1-2x_2+x_3 &= 0 \\ 2x_1-x_2-x_4 &= 0 \\ x_1+x_2+x_3+x_4+x_5 &= 1 \\ x_1,x_2,x_3,x_4,x_5 &\ge 0 \end{aligned}$$

y

$$\min\ 2^{22}x_1+2^{22}x_2+\lambda$$

$$\begin{aligned} s.a.\ \ (-1+2^{22})x_1+(-1+2^{22})x_2+(-1-2^{22})x_3-x_4-x_5+(5-2^{22})\lambda &= 0 \\ (-1+2^{22})x_1+(-1-2^{22})x_2-x_3+(-1+2^{22})x_4-x_5+(5-2^{22})\lambda &= 0 \\ x_1+x_2+x_3+x_4+x_5+\lambda &= 1 \\ x_1, x_2, x_3, x_4, x_5, \lambda &\ge 0 \end{aligned}$$

Al aplicar el algoritmo de Karmarkar, para ambos problemas utilizamos $\alpha = 0.2$ que corresponde a $\delta = 0.15$, como parámetro. Para definir el criterio difuso utilizaremos tres tipos de funciones de pertenencia, lineal, no lineal cóncava y no lineal convexa.

Para resolver el primer problema, utilizaremos como función de pertenencia en el criterio de parada difuso, por un lado la función lineal y por otro lado la función no lineal de la forma (2.39) con $r = 1/8$ y $c_m = 53687091.2$ obtenido en el paso inicial del algoritmo.

Los resultados obtenidos resumidos en la Tabla 2.2, nos muestra de manera muy clara una mejor aproximación cuando se utiliza la función no lineal.

α	Función Lineal			Función no lineal		
	$g^{-1}(\alpha)$	Costo (aprox.)	Iteración	$g^{-1}(\alpha)$	Costo (aprox.)	Iteración
0.977	1234800	1145621	25	4.204289×10^{-6}	3.614893×10^{-6}	151
0.9976	128850	115176	36	5.909623×10^{-14}	5.176798×10^{-14}	237

Tabla 2.2: Costos aproximados al óptimo 0, del ejemplo 1.

α	Función de pertenencia Lineal			Función de pertenencia no lineal		
	$g^{-1}(\alpha)$	Costo (aprox.)	Iteración	$g^{-1}(\alpha)$	Costo (aprox.)	Iteración
$1\text{-}10^{-7}$	0.1398	No factible		442.119	418.7245	48
$1\text{-}10^{-12}$	0.00001			1.39809	1.397536	9995

Tabla 2.3: Grados de aproximación al óptimo 1, del ejemplo 2.

Para el segundo problema igualmente se utiliza el caso lineal, y el caso no lineal (2.39) pero esta vez con r=2. El costo inicial es c_m=1398101.5. Como se observa en la Tabla

2.3, al utilizar $\alpha=1-10^{-7}$ se obtiene $g^1(\alpha)$ menor que el costo mínimo óptimo 1 cuando g es lineal mientras que para g no lineal es mayor que 1. Este último resultado es más beneficioso para obtener la solución aproximada aunque perjudica la calidad de la aproximación.

En los dos ejemplos se observa que para tener una solución aproximada aceptable no ha sido necesario realizar todas las iteraciones, que eran 6067 en el primer caso, y 20960 en el segundo caso.

2.4 CRITERIOS DE PARADA DIFUSOS EN ALGORITMOS MODIFICADOS DE KARMARKAR

2.4.1 Introducción

Desde que Karmarkar publicó en 1984, el método de solución de PL que lleva su nombre, se han propuesto muchas modificaciones en diferentes aspectos.

Como se ha visto el algoritmo de Karmarkar resuelve un problema de minimización de costo no negativo con restricciones de igual a cero y que la suma de las variables sea igual a la unidad, este problema es denominado **forma estándar** de Karmarkar. Esta característica permite elegir como solución inicial factible el punto *e/n*, donde *e* es el vector con todos sus componentes igual a 1, el siguiente punto determina en la dirección del gradiente descendiente que se encuentra dentro de la región factible y con el menor costo. Durante el proceso de iteración, el algoritmo de Karmarkar, mantiene esta situación para ello utiliza una transformación, denominada **transformación proyectiva**, de tal manera que el nuevo punto obtenido sea trasformado en *e/n*, con lo cual se regresa a la situación del punto inicial para repetir el proceso. Para analizar la **convergencia** de la sucesión de puntos que obtiene de esta forma utiliza una función potencial. El algoritmo de Karmarkar, originalmente propuesto, garantiza la convergencia de la solución cuando el costo óptimo es cero, y en el caso del costo óptimo distinto de cero sólo informa de ello pero no garantiza la convergencia, aunque esta limitación no es ningún inconveniente para resolver cualquier problema mediante el algoritmo de Karmarkar, porque siempre es posible transformarlo

a la forma estándar, esta limitación lo transforma en algo más complejo al problema.

Frente a estas limitaciones y con el afán de reducir el total de las operaciones, el número de iteraciones en suma para mejorar la eficiencia han aparecido muchas modificaciones en diferentes aspectos del algoritmo. A manera de ejemplo podemos citar algunas modificaciones propuestas. Barnes [9] y Vanderbei y otros [200] presentan el algoritmo modificado de Karmarkar para resolver un problema de minimización general y sin la restricción de igual a cero y sin la condición de la suma igual 1 o igual a n. Barnes [9] y Kojima [104] presentan la variación del algoritmo para determinar las variables básicas de la solución óptima. Goldfard y Mehrotra [74,75] utilizan una proyección inexacta en la transformación proyectiva y la dirección del gradiente reducido en lugar del gradiente decreciente. Gay [70], Todd y Burrell [194] y Ye y Kojima [214] proponen una modificación para resolver el problema estándar de Karmarkar pero con funciones objetivos generales, no necesariamente con solución óptima cero y a su vez proporcionan la solución óptima del problema dual. Vanderbei [199] presenta las modificaciones del algoritmo de Karmarkar para resolver problemas de PL en su forma estándar pero con variables sin restricciones de no negatividad.

Las modificaciones propuestas recogidas aquí son fundamentalmente aquellas que disminuyen el número de iteraciones o el número de operaciones.

Como sabemos todo problema de PL puede expresarse en la forma

$$\begin{aligned} &\min cx \\ &\text{s.a. } Ax = b \\ &\quad x \geq 0 \end{aligned} \tag{2.40}$$

Sea $F = \{ x \in R^n / x \geq 0, Ax = b \}$ la región factible, si F es no vacía, entonces el problema (2.40) es factible. Si $\min_{x \in \Omega} cx > -\infty$, el problema (2.40) es limitado.

En el caso de que el problema (2.40) no tenga soluciones factibles degeneradas, el conjunto $F^o = \{ x \in R^n / x > 0, Ax = b \}$ es el interior de F relativo al espacio afín $\{ x \in R^n /, Ax = b\}$.

Vanderbei, Meketon y Freedman [200] consideran que no es necesario transformar el problema (2.40) en la forma estándar de Karmarkar para aplicar el algoritmo de Karmarkar, simplemente es necesario hacer una pequeña modificación de la siguiente manera:

$$\begin{aligned} &\min cx \\ &\text{s.a. } A'x = b \\ &\quad x \geq 0 \end{aligned} \tag{2.41}$$

donde

$$A' = [A \vdots b\text{-}Ae^T],$$

es una matriz de orden mx(n+1), x y c son (n+1)-vectores con $c_{n+1} = M$ es un costo muy alto que asegura que la variable x_{n+1} tendrá valor cero en el óptimo.

2.4.2 Modificación propuesta por Vanderbei, Meketon y Freedman

Para aplicar el algoritmo de Karmarkar en el problema (2.41), Vanderbei, Meketon y Freedman [200] hacen pequeñas modificaciones, fundamentalmente en la definición de la transformación y en la regla de parada. Así obtienen el siguiente algoritmo, denominémoslo *Algoritmo de Karmarkar-VMF (K-VMF)*:

Etapa inicial: $x^0 = e$, y sea ε, el nivel de aproximación de deseada, $e=(1,1,...,1)$ n-vector.

Etapa iterativa: Calcular $x^{k+1} = T(x^k)$

donde

$T(x) = x - (\alpha/\gamma)D_x^2 r$

$D_x = \text{diag}(x)$

$r = r(x) = c\text{-}A^T w$ (vector de costos reducidos correspondientes a x)

$w = w(x) = (AD_x^2 A^T)^{-1} AD_x^2 c$ (Vector de variables duales correspondientes a x)

$\alpha \in (0,1)$ es una constante y

$\gamma = \gamma(x) = \max_i (x_i . r_i(x))$.

Criterio de parada: Si $d = n(\gamma(x^k)+\delta(x^k)M) \leq \varepsilon$, detener el proceso y x^k es la solución con costo cx^k cercana al óptimo con una aproximación ε deseada, donde:

$$\delta(x^k) = -\min_i r_i(x^k) \quad \text{y} \quad M = e.x^k/n.$$

Los autores demuestran la convergencia de la sucesión de puntos generados por este algoritmo para el caso de que tanto el primal como el dual del problema (2.40) no tengan soluciones degeneradas, es decir si AD_x tiene rango máximo (igual al número de filas) para todo $x \in \Omega$ y $r(x) = c\text{-}A^T w$ tiene a lo sumo m ceros para todo $w \in R^m$, respectivamente.

Otra característica importante de este algoritmo modificado que han propuesto Vanderbei, Meketon y Freedman es la que proporciona la solución del problema dual. Con esta misma característica proponen también Todd y Burrell [194] un algoritmo modificado, pero siempre manteniendo la forma estándar de Karmarkar. El algoritmo propuesto por Todd y Burrell es un poco más eficiente que el algoritmo inicial de Karmarkar.

Vanderbei, Meketon y Freedman demuestran la convergencia de la sucesión de las soluciones que genera su algoritmo, sin embargo no precisan el número de iteraciones necesarias para alcanzar la solución óptima, por tanto se entiende que la condición de optimalidad es $d = 0$. Teniendo en cuenta que para alcanzar esta condición de optimalidad se necesitaría un número muy elevado de iteraciones, los mismos autores proponen una determinada aproximación, mediante una condición de parada en el sentido de que d tenga un valor muy pequeño que se aproxima a cero tanto como se quiera. Esta condición, que expresada mucho mejor con el término lingüístico "casi cero", cuya representación simbólica, aunque los autores no lo hayan percibido así, es el número difuso $d =_f 0$, definido mediante la función de pertenencia dada en (2.37) en donde en lugar de c_m en este caso se usará d_0 obtenido en el paso inicial. Entonces si se fija un nivel de tolerancia α para el cumplimiento de la condición de parada, el **criterio de parada difuso** queda expresada de la siguiente manera:

$$d \leq g^{-1}(\alpha) \tag{2.42}$$

Particularmente, por su simplicidad, se sugiere utilizar la función g como la dada en (2.39), con d_0 en lugar de c_m y $0<r\leq 1$.

Para ilustrar el criterio de parada difuso en el algoritmo K-VMF utilizamos los mismos ejemplos de la sección anterior 2.3, para ambos ejemplos usamos dos funciones de pertenencia, una lineal y otra no lineal en este último la función definida por (2.39) con r=1/8. En el primer ejemplo $d_0 = 2.4$ y en el segundo ejemplo $d_0 = 2.666...$ Los resultados se muestran en las Tablas 2.4 y 2.5 respectivamente.

α	Función de pertenencia lineal			Función de pertenencia no lineal		
	$g^{-1}(\alpha)$	Costo (aprox.)	Itera.	$g^{-1}(\alpha)$	Costo (aprox.)	Itera.
0.977	0.0552	0.0110454	22	$1.879\text{x}10^{-13}$	$3.410795\text{x}10^{-14}$	126
0.9976	0.00576	0.00111403	31	$2.6418\text{x}10^{-21}$	$3.915506\text{x}10^{-16}$	267

Tabla 2.4: Costos aproximados al óptimo 0, del ejemplo 1 de la sección 2.3.

α	Función de pertenencia lineal			Función de pertenencia no lineal		
	$g^{-1}(\alpha)$	Costo (aprox.)	Itera	$g^{-1}(\alpha)$	Costo (aprox.)	Itera
0.977	$2.6667\text{x}10^{-7}$	1.0000000039	15	$2.6667\text{x}10^{-26}$	1.0000000039	59
0.9976	$2.6667\text{x}10^{-12}$	1.0000000039	70	$2.6667\text{x}10^{-56}$	1.0000000039	99

Tabla 2.5: Grados de aproximación al óptimo 1, del ejemplo 2 de la sección 2.3.

2.4.3 Modificación propuesta por Barnes

Es bien conocido que resolver un problema de la forma (2.40) consiste en determinar una submatriz B de A tal que $x^*=B^{-1}b$ sea la solución básica óptima. Barnes [9] presenta una variación del algoritmo de Karmarkar que permite identificar las variables básicas factibles óptimas, es decir, identificar la matriz básica óptima B. Notemos por K-Barnes el algoritmo modificado de Karmarkar propuesto por Barnes, para resolver el problema (2.41), y es el siguiente:

Algoritmo K-Barnes:

Etapa inicial: $x^0 = e$, $e = (1,1,...,1)$ n-vector, $\varepsilon > 0$ pequeño..

Etapa iterativa: Conociendo:

$$D_k = diag\,(x^k)$$

$$y_k = (AD_k^2 A^T)^{-1} AD_k c$$

$$R = \min_{z_i^k > 0} \frac{\left\| D_k (z^k) \right\|}{x_i^k z_i} - \alpha, \quad 0 < \alpha < 1,$$

$$z^k = c - A^T y_k$$

Calcular:

$$x^{k+1} = x^k - R \frac{D_k^2 (c - A^T y_k)}{\left\| D_k (c - A^T y_k) \right\|} \tag{2.43}$$

Criterio de parada: m componentes de z^k pertenecen al intervalo $(-\varepsilon,\varepsilon)$.

Barnes demuestra que si el problema no tiene soluciones básicas degeneradas, la sucesión z^k converge a un punto con m componentes igual a cero y los demás positivos. Las componentes con valor cero indican las variables básicas. Una propuesta algo parecida presenta Kojima [104], que consiste en determinar las variables básicas durante el proceso de iteración del algoritmo de Karmarkar, las mismas que excluye en las iteraciones siguientes, simplificando de esta manera el problema inicial.

En este algoritmo K-Barnes, para obtener la solución exacta el número de iteraciones debería ser un número suficiente de tal manera que permita identificar apropiadamente las variables básicas óptimas. El autor no propone una cota general para el valor de ε que puede ser adecuado en todos los problemas, y debido a que no existe ningún indicador para garantizar que se obtienen una solución exacta, las soluciones obtenidas con $\varepsilon>0$ siempre serán soluciones aproximadas. La solución exacta se obtendrá sólo cuando $\varepsilon=0$.

Frente a estas condiciones, con la introducción de la regla de parada difusa se pretende proporcionar el grado en que la solución aproximada obtenida se acerca a la solución óptima. Porque el valor de ε que se utiliza estará asociado en cierto modo con la solución inicial.

Sea δ^k el máximo valor de entre las m menores componentes de $|z^k|$, donde $|.|$

representa el valor absoluto de las componentes respectivamente. Sea δ^0 el valor obtenido en la solución inicial, si $\delta^0=0$, la solución inicial proporciona la solución óptima. En caso contrario el criterio de parada puede ser expresado como "δ^k casi cero" expresión lingüística definida por la función de pertenencia (2.37), en donde c_m es sustituido por δ^0, así la condición de parada que proporciona la solución con un nivel de tolerancia α será la siguiente:

Criterio de parada difuso: $\delta^k < g^{-1}(\alpha)$

Igual que en los casos anteriores se recomienda utilizar funciones de la forma (2.39), con $c_m = \delta^0$ y $0 < r \leq 1$.

En los mismos ejemplos de la sección 2.3 se ha utilizado este algoritmo, con $\delta^0=0.9$ para el primer ejemplo y $\delta^0=1/3$ para el segundo ejemplo. Para los dos ejemplos, en los criterios de parada difusos, se utiliza por un lado la función lineal y por otro lado la función no lineal (2.39) con r=1/8. Los resultados se muestran en las Tablas 2.6 y 2.7 respectivamente.

α	Función de pertenencia lineal			Función de pertenencia no lineal		
	$g^{-1}(\alpha)$	Costo	Iteración	$g^{-1}(\alpha)$	Costo	Iteración
0.977	0.0207	0	25	7.0479×10^{-14}	0	34
0.9976	0.00216	0	26	9.9067×10^{-22}	0	39

Tabla 2.6: Costos aproximados al óptimo 0, del ejemplo 1 de la sección 2.3.

α	Función de pertenencia lineal			Función de pertenencia no lineal		
	$g^{-1}(\alpha)$	Costo	Iteración	$g^{-1}(\alpha)$	Costo	Iteración
0.977	0.00766	1	1	2.61036×10^{-14}	1	6
0.9976	0.000799	1	1	3.6691×10^{-22}	1	8

Tabla 2.7: Grados de aproximación al óptimo 1, del ejemplo 2 de la sección 2.3.

2.5 CRITERIOS DE PARADA DIFUSOS EN ALGORITMOS DE PUNTOS INTERIORES

2.5.1 Introducción

Los algoritmos de puntos interiores denominados también como algoritmos de escala afín (affine-scaling) son obtenidos mediante la unificación de diversas variaciones del algoritmo de Karmarkar. El algoritmo de Karmarkar, por ser un algoritmo que no utiliza la característica combinatorial de los problemas de PL y con tiempo de ejecución polinomial con respecto a la dimensión del problema, es un digno competidor del algoritmo simplex en la solución de problemas de PL. Desde que aquél [99] apareció en 1984, ha sido y sigue siendo hasta ahora punto de atención de muchos investigadores en la materia. Desde el punto de vista aplicativo, los dos grandes inconvenientes del algoritmo de Karmarkar se presentaron en la forma inicial de la resolución del problema y en la función objetivo, la cual se transformaba en no lineal después de aplicar la transformación proyectiva en cada iteración.

Como se ha visto, para aplicar el algoritmo de Karmarkar, todo problema de PL debe ser de mínimización con función objetivo no negativa, además de que las restricciones lineales deben ser iguales a cero y la suma de todas las variables igual a uno. Un problema de la forma (2.40), para ser resuelto mediante el algoritmo de Karmarkar debería ser trasformado en la forma (2.25'). La desventaja de esto radica en el incremento de la dimensión del problema, como podemos apreciar en el ejemplo (2.36), de dimensión 262, mientras que el problema inicial es de dimensión 20. Y si la función objetivo no fuera no negativa la dimensión se hubiera incrementado a 524.

El algoritmo de Karmarkar en cada iteración obtiene una segunda solución a partir de una solución conocida. Para ello, se construye un nuevo problema estándar de Karmarkar mediante una transformación proyectiva. Esta proyección transforma a su vez a la función objetivo en una función no lineal con lo cual el nuevo problema queda fuera del contexto de PL. Este inconveniente es parcialmente superado con la introducción de una función potencial, aunque con las consecuentes limitaciones en la obtención de la

convergencia en el caso de que el mínimo no sea cero.

Como hemos visto en la sección anterior, Barnes [9], Gay [70], Vanderbei [199] y Vanderbei, Meketon y Freedman [200] han introducido la transformación lineal en lugar de la proyectiva. Bayer y Lagarias [10], De Gellinck y Vial [71], Iri y Imai [95], Nazareth [146], Rinaldi [159], Todd y Ye [195] han introducido en la PL los métodos clásicos de la programación no lineal. Asimismo, Lustig [129], Lustig, Marsten y Shanno [130], Monma y Morton [139], Monteiro y Adler [140], Monteiro, Adler y Resende [141], entre muchos otros, han contribuido en el desarrollo de lo que se ha dado en llamar *algoritmos de puntos interiores* de la PL, algoritmos que han sido sistematizados por Arbel [3], sobre cuyo trabajo nos basamos aquí.

En la siguiente sección 2.5.2 presentamos las ideas geométricas y sus respectivas representaciones algebraicas de los algoritmos de puntos interiores, en la sección 2.5.3, los algoritmos en sí y finalmente, en la sección 2.5.4 introducimos las reglas de parada difusas en dichos algoritmos.

2.5.2 Ideas básicas

Como en todo algoritmo, debemos precisar los tres pasos fundamentales de los algoritmos de puntos interiores que son:

i) Etapa inicial

ii) Etapa iterativa

iii) Criterio de parada.

Cada uno de estos pasos se aplicarán al PLE (2.40). Esta forma es obtenida a partir de otras formas incluyendo variables de holgura y/o artificiales.

Etapa inicial

El procedimiento inicial consiste en elegir un punto interior de la región factible, es decir, un vector que satisface todas las restricciones del problema a resolver.

En general, cualquier vector puede ser considerado como un punto interior inicial x^0,

ya que si no satisface las restricciones, siguiendo la sugerencia de Vanderbei y otros [200], se puede introducir una nueva variable con costo muy alto M y con coeficientes $b\text{-}x^0A$ en las restricciones, obteniendo así un problema de la forma (2.41). Naturalmente esta modificación incrementa el tamaño del problema con la consiguiente ralentización de la convergencia de la solución. Para evitar este inconveniente nosotros proponemos utilizar el algoritmo elipsoidal para determinar una solución factible inicial del problema en su forma canónica.

Etapa iterativa

Conociendo un punto interior factible x^0, el paso iterativo permite generar un nuevo punto factible que reduzca el costo actual.

Puesto que se desea minimizar una función dentro de la región factible, la dirección del gradiente descendente nos proporciona los puntos de mejor decrecimiento. Por tanto, para obtener el nuevo punto debemos desplazarnos en dicha dirección, y la longitud del desplazamiento debe ser de tal manera que el nuevo punto sea punto interior factible. Además, este desplazamiento debe ser lo más grande posible para alcanzar el punto óptimo con el menor número de pasos. Para compatibilizar estas dos condiciones se utiliza una circunferencia con centro en el punto conocido e inscrita en la región factible. Un desplazamiento de longitud ligeramente menor que el radio garantiza la factibilidad.

Por otro lado, si el punto conocido es el centro de la región factible, el desplazamiento será mayor que en el caso contrario, y se acercará mejor al punto óptimo. Esta situación se logra parcialmente haciendo un cambio de escala en el sistema de coordenadas.

Si x^0 es el punto factible conocido, la transformación $x'=Dx$, con $D=diag(x^0)$ hace que el punto factible conocido x^0 se transforme en $e = (1,1,...,1)$, punto factible del problema escalado:

$$\begin{aligned} &\min c'x' \\ &\text{s.a. } A'x' = b \\ &\quad x' \geq 0 \end{aligned} \tag{2.44}$$

donde $A' = AD$, $c' = Dc$.

Si $x'^1 = x'^0 + dx'$ es el nuevo punto, donde dx' es el vector dirección de paso, la condición de factibilidad requiere que $Ax'^1 = A(x'^0 + dx') = Ax'^0 + Adx'^1 = b$, pero como $Ax'^0 = b$, entonces $Adx' = 0$; esto último significa que el vector dirección dx debe pertenecer al espacio nulo de la matriz A. Puesto que dicho vector sigue la dirección del gradiente descendente de la función objetivo, este gradiente descendente $-c$ debe ser proyectado ortogonalmente al espacio nulo de A' mediante el operador proyección $P=I_n-A'^T(A'A'^T)^{-1}A'$, es decir, $dx'=-P'c'$. Finalmente, para mantener la condición de punto interior el nuevo punto estaría dado por

$$x'^1 = x'^0 + \alpha dx' \tag{2.45}$$

con $\alpha<1$.

Este nuevo punto x'^1 debe ser expresado en términos del sistema de coordenadas iniciales. Esto se logra mediante una transformación afín. Debido a que se utiliza el proceso de cambio de escala mediante una transformación afín, se denominan también algoritmos "affine-scaling".

De lo expresado arriba se tiene:

$$dx' = -P'c' = (I_n-A'^T(A'A'^T)^{-1}A')\ c' = -D[c-A^T(AD^2A^T)^{-1}AD^2c]$$

haciendo

$$(AD^2A^T)y = AD^2c \ \rightarrow\ y = (AD^2A^T)^{-1}AD^2c \tag{2.46}$$

entonces $dx' = -D[c-A^Ty]$ luego $dx = Ddx' = -D^2[c-A^Ty]$ y si hacemos

$$z = c-A^Ty \tag{2.47}$$

entonces

$$dx = -D^2z \tag{2.48}$$

finalmente

$$x = x^0 + \alpha dx \tag{2.49}$$

es el nuevo punto factible con menor costo.

Sabemos que todo problema de PL tiene su respectivo problema dual. Una de las ventajas del procedimiento que acabamos de explicar radica en el hecho de que en cada iteración se obtiene una solución aproximada del problema primal, así como también para el problema dual. Si el x dado en (2.49) es el nuevo punto factible o solución aproximada,

entonces el y dado en (2.46) es el punto factible del problema dual.

Criterio de parada

Dados los problemas primal (P) y su respectivo dual (D)

$$(P)\quad \begin{array}{l}\min c^T x \\ \text{s.a. } Ax = b \\ x \geq 0\end{array} \qquad (D)\quad \begin{array}{l}\max b^T y \\ \text{s.a. } A^t y \leq c \\ y \text{ irrestrictas}\end{array} \qquad (2.50)$$

por el teorema de la dualidad débil se tiene que $cx \geq b^T y$, y por el teorema general de la dualidad en el óptimo se cumple la igualdad. Por lo tanto para alcanzar la solución óptima del problema primal y dual el procedimiento debe detenerse cuando se obtienen x^*, y^* tales que $cx^* = b^T y^*$. Sin embargo, debido a que en cada iteración se obtienen puntos interiores mientras que el punto óptimo es de la frontera, el procedimiento no determina exactamente el punto óptimo sino una sucesión de puntos que convergen al punto óptimo. En consecuencia, el procedimiento se realiza hasta alcanzar un valor fijado como un umbral para la diferencia entre los valores de las funciones objetivos primal y dual.

El criterio de parada que han sugerido muchos autores es:

$$\text{gap} = \|cx - b^T y\| / (1 + \|cx\|) \leq \varepsilon \qquad (2.51)$$

donde ε varía entre 10^{-6} y 10^{-8}.

2.5.3 Algoritmos de puntos interiores

En el algoritmo que se ha explicado en la sección anterior, aunque en cada iteración se obtiene una solución aproximada x para el problema primal, y otra solución aproximada y para el problema dual, el procedimiento mantiene que la solución aproximada del problema primal sea factible, por lo que se denomina *algoritmo primal* . Si en lugar de mantener la factibilidad primal se modifica el procedimiento de tal manera que mantenga la factibilidad dual, se obtendría un algoritmo similar denominado *algoritmo dual.* Si se mantiene la factibilidad primal y dual simultáneamente, se obtiene el algoritmo denominado *algoritmo primal-dual.*

2.5.3.1 Algoritmo Primal

Este algoritmo recoge fundamentalmente las aportaciones de Barnes [9], Vanderbei [199] y Vanderbei, Meketon y Freedman [200] entre otros, cuyo procedimiento se ha explicado en la sección anterior, por lo tanto sin más comentario, presentamos en forma ordenada el algoritmo primal, para resolver los problemas primal y dual dados en (2.50):

Etapa Inicial:

Paso 0: Dado $x_0 > 0$ un punto interior factible, hacer k=0, y $x(k)=x_0$.

Etapa iterativa:

Paso 1: Definir: D(k)=diag(x(k)), la matriz de cambio de escala.

Paso 2: Calcular la estimación dual y(k), resolviendo $[AD^2(k)A^T]y(k)=AD^2c$

Paso 3: Evaluar el vector costo reducido $z(k)=c-A^Ty(k)$, luego $dx(k) = -D^2(k)z(k)$

Paso 4: El siguiente vector solución está dado por:

$$x(k+1) = x(k) + \rho\alpha dx(k)$$

donde $$\alpha = \min\left\{-\frac{x_i(k)}{dx_i(k)}: \quad \forall\, dx_i(k) < 0, 1 \le i \le n\right\} \qquad \text{y} \quad 0 < \rho < 1$$

Criterio de Parada:

Paso 5: Si el criterio de parada se cumple, terminar el proceso. En el caso contrario incrementar k e ir al paso 1.

2.5.3.2 Algoritmo Dual

Este algoritmo mantiene la factibilidad dual, es decir, a partir de un punto factible conocido del problema dual determina otro punto factible de tal manera que el valor de la función objetivo sea mayor. Las propuestas de Todd y Burrell [194] y de Monma y Morton [139] son las aportaciones de mayor relevancia en este algoritmo.

Introduciendo variables artificiales en el problema dual de (2.50) se obtiene el

problema estándar siguiente:

$$\begin{aligned} &\max\ b^T y \\ &\text{s.a.} \quad A^t y + z = c \\ &\quad y \text{ irrestrictas},\ z \geq 0 \end{aligned} \tag{2.52}$$

Asumiendo que un punto factible inicial (y^0, z^0) de (2.52), con $z^0 \geq 0$ es conocido, el siguiente paso consiste en determinar los vectores dirección de paso dy y dz de tal manera que nos permita determinar (y,z) un nuevo punto interior factible de (2.52), donde $y = y^0 + \alpha\, dy$, y irrestrictas; $z = z^0 + \alpha dz$, $z \geq 0$. De las condiciones de factibilidad se deduce que $A^T dy + dz = 0$; esta dependencia nos permite determinar una de las direcciones a partir de la otra. La condición de no negatividad de z nos permite hacer los cálculos de una manera semejante que en el algoritmo primal.

Los pasos del algoritmo dual son los siguientes:

Etapa inicial:

Paso 0: Dados y^0, z^0 vectores iniciales factibles, con $z^0 > 0$, sean $k = 0$, $y(k) = y^0$ y $z(k) = z^0$.

Etapa iterativa:

Paso 1: Definir $D(k) = \text{diag}[1/z_1(k), 1/z_2(k), \ldots, 1/z_n(k)]$.

Paso 2: Hallar $dy(k)$, resolviendo el sistema $[AD^2(k)A^T]dy(k) = b$.

Paso 3: Calcular: $dz(k) = -A^t dy(k)$.

Paso 4: Encontrar el siguiente costo reducido: $z(k+1) = z(k) + \rho\alpha dz$

el vector dual $y(k+1) = y(k) + \rho\alpha dy(k)$,

donde $\alpha = \min\left\{-\dfrac{z_i(k)}{dz_i(k)} : \ \forall\, dz_i(k) < 0, 1 \leq i \leq n\right\}$ y $0 < \rho < 1$

y el correspondiente vector primal $x(k+1) = -D^2 dz(k)$.

Criterio de parada:

Paso 5: Si el criterio de parada se cumple, terminar las iteraciones. En el caso contrario incrementar k en una unidad e ir al paso 1.

2.5.3.3 Algoritmo Primal-Dual

Como ya hemos comentado al inicio de la sección (2.5.3), el algoritmo primal-dual mantiene la factibilidad primal y dual al mismo tiempo, en cada iteración. Para poder hacer esto, las condiciones de no negatividad tanto del problema primal como del dual son expresadas en otra forma, por tanto, el problema primal de (2.50) y el dual (2.52) son transformados respectivamente en:

$$(P_\mu): \quad \min \quad c^T x - \mu \sum_{i=1}^{n} \ln(x_i) \qquad (2.53)$$
$$\text{s.a.} \quad Ax = b, \quad \mu > 0.$$

$$(D_\mu): \quad \max \quad b^T y + \mu \sum_{i=1}^{n} \ln(z_i) \qquad (2.54)$$
$$\text{s.a.} \quad A^T y + z = c, \quad \mu > 0.$$

Los términos que contienen la función logaritmo no sólo obligan a las variables a ser positivas sino a mantenerse en una posición central con respecto a los ejes para alcanzar la optimalidad. El parámetro de *barrera* μ sirve para mantener la posición centrada de los puntos interiores y al mismo tiempo la condición de reducción del costo.

Como podemos observar, los problemas (2.53) y (2.54) son no lineales, por tanto, debemos aplicar los métodos para resolver problemas de optimización no lineales. Las propuestas de Bayer y Legarias [10], De Gellinck y Vial [71], Iri e Imai [95], Lustig [129] y, Montero y Adler [140] entre otros, han permitido desarrollar el algoritmo Primal-Dual, cuyos pasos resumimos a continuación:

Etapa Inicial:

Paso 0: Sean x^0, y^0 y z^0, puntos interiores factibles, es decir, que satisfacen las restricciones de (2.53) y (2.54), con $x^0>0$ y $z^0>0$. Hacer k=0 y $x(k)=x^0$, $y(k)=y^0$ y $z(k)= z^0$.

Etapa Iterativo:

Paso 1: Evaluar el parámetro barrera: $\mu=0.1z^T(k)x(k)/n$.

Paso 2: Definir las matrices diagonales de cambio de escala siguientes:

$X(k) = \text{diag}[x(k)]$, $Z(k) = \text{diag}[z(k)]$ y $D^2(k)=Z^{-1}X$.

Paso 3: Evaluar el vector auxiliar: $v(\mu)=\mu e-XZe$, $e=(1,1,...,1)^T$.

Paso 4: Hallar el vector dirección de paso dual dy resolviendo el sistema:

$$[AD^2(k)A^T]dy(k) = -AZ^{-1}v(\mu)$$

Paso 5: Evaluar los vectores dirección de paso:

$$dz(k)=-A^T dy(k) \quad y \quad dx(k)=[Z^{-1} - D^2A^T(AD^2A^T)^{-1}AZ^{-1}]v(\mu)$$

Paso 6: Evaluar:

$x(k+1) = x(k) + \rho\alpha_p dx(k)$, nuevo punto interior primal,

$y(k+1) = y(k) + \rho\alpha_d dy(k)$, nuevo punto interior dual y

$z(k+1) = z(k) + \rho\alpha_d dz(k)$, nuevo costo reducido.

Donde:

$$\alpha_p = \text{mín}\left\{-\frac{x_i(k)}{dx_i(k)}: \quad \forall\, dx_i(k)<0, 1\le i\le n\right\}$$

$$\alpha_d = \text{mín}\left\{-\frac{z_i(k)}{dz_i(k)}: \quad \forall\, dz_i(k)<0, 1\le i\le n\right\}$$

Criterio de parada:

Paso 7: Si el criterio de parada se cumple finalizar la iteración, en el caso contrario aumentar en uno el valor de k e ir al paso 1.

2.5.4 Criterio de parada difuso en algoritmos de puntos interiores

El criterio de parada en los algoritmos de puntos interiores expuestos en la sección 2.5.3 presenta una aproximación a la solución óptima que en muchos casos puede alcanzarse con un número muy elevado de iteraciones.

Para obtener una solución exacta la condición de parada matemáticamente exacta en estos algoritmos sería:

$$\lambda = \text{gap} = \|cx\text{-}b^T y\|/(1+\|cx\|)=0. \tag{2.55}$$

Sin embargo debido a que estos algoritmos son de aproximación no es práctico imponer esta condición de parada, por ello se ha establecido la condición (2.51). Esto nos dice que la solución óptima, es decir, el punto de convergencia se puede obtener cuando λ

se encuentra entre 10^{-6} y 10^{-8}. Evidentemente, otros umbrales inferiores a esos números posiblemente proporcionarían soluciones más cercanas al óptimo pero no óptimas. La diferencia entre el punto óptimo y la solución aproximada será menor cuanto menor sea el umbral para λ.

El criterio de parada (2.51) obedece a un criterio de aproximación fundamentalmente de precisión, y no es susceptible de ser reajustado en función a las características de los problemas a resolver, ni en función a las expectativas del decisor, pese a que este último es quién en última instancia las ha de usar.

Siguiendo con el objetivo de este trabajo, también proponemos flexibilizar los algoritmos de puntos interiores, de tal manera que sea posible controlar la detención del proceso de cálculo en función al tiempo de ejecución razonable, y que el resultado obtenido sea aceptable para el decisor aunque no sea la óptima.

Puesto que la condición de parada (2.55) (λ=0) es similar a la condición de parada del algoritmo de Karmarkar, el criterio de parada difuso estaría dado por la expresión lingüística "λ positivamente casi cero" es decir un número difuso con función de pertenencia dado por (2.37). Igualmente la función g es la función definida por (2.39) con $0<r<1$ y c_m se sustituye por λ_0 (calculado según (2.51) o (2.55)) obtenido en la primera iteración.

Una vez introducida la condición de parada difusa, el decisor podrá fijar el grado α ($0<\alpha<1$) de tolerancia del incumplimiento de la regla de parada (2.55), es decir, el grado de cercanía a cero de λ para considerar aceptable la solución que se obtenga. Con lo cual se tiene el ***Criterio de Parada difuso:***

$$\lambda \leq g^{-1}(\alpha)$$

esta condición debe ser utilizada en cada uno de los criterios de parada de los tres algoritmos de puntos interiores, así se obtendrán dichos algoritmos con criterios de parada difuso.

α	**Algoritmo Primal** $x^0=(2,2,2,2)$ $\lambda_0=0.6505$				**Algoritmo Dual** $y^0=(-1,3/2)$ $z^0=(1,1/2,1,3/2)$ $\lambda_0=0$				**Algoritmo Primal-Dual** $\lambda_0=0.3668$			
	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	iteración	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	iteración	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	iteración
Función de pertenencia lineal $g(t)=1-t/\lambda_0$												
0.977	0.015	0.01066	0	4	0	0	0	1	0.0084	0.00202	0	3
0.9976	0.00156	0.00152	0	25	0	0	0	1	0.00088	0.000112	0	4
Función de pertenencia no lineal $g(t)=1-(t/\lambda_0)^{1/8}$												
0.5	0.00254	0.002447446	0	15	0	0	0	1	0.0014	0.000112	0	4
0.8	1.6654×10^{-6}	1.665314×10^{-6}	0	25274	0	0	0	1	0.0000009	0.00000034	0	6
0.9	6.5054×10^{-9}	6.505376×10^{-9}	0	6472365	0	0	0	1	3.66×10^{-9}	1.033498×10^{-9}	0	8

Tabla 2.8: Grados de aproximación al costo óptimo 0 del ejemplo 1 de la sección 2.3.

α	**Algoritmo Primal** $x^0=(3,3,5,1)$ $\lambda_0=0.3695$				**Algoritmo Dual** $y^0=(1/2,-1/4)$ $z^0=(3/4,1/4,1/2,1/4)$ $\lambda_0=0.1001$				**Algoritmo Primal-Dual** $\lambda_0=0.2681$			
	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	itera-ción	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	iteración	$g^{-1}(\alpha)$	Costo Primal	Costo Dual	itera-ción
Función de pertenencia lineal $g(t)=1-t/\lambda_0$												
0.977	0.0085	1.000625	0.999695	3	0.003	0.999991	0.9960991	5	0.0062	1.000625	0.99514	3
0.9976	0.0008868	0.000625	0.999695	3	0.00024	0.9999999	0.99952	59	0.00064	1.000031	0.99964	4
Función de pertenencia no lineal $g(t)=1-(t/\lambda_0)^{1/8}$												
0.5	0.0014	1.000625	0.9996953	3	0.00039	0.9999998	0.99923	35	0.001	1.000031	0.99964	4
0.8	9.4594×10^{-7}	1.0000018916	$1-7.16\times10^{-12}$	5545	0.00000025	$1-7\times10^{-14}$	0.9999995	61602	0.0000007	1.0000000039	0.9999999916	7
0.9	3.6954×10^{-9}	1.0000000036	1.00	1310853	1.001×10^{-9}	1.00	0.9999999992	15923095	2.68×10^{-9}	$1.0+1.9\times10^{-11}$	0.999999999953	8

Tabla 2.9: Grados de aproximación al costo óptimo 1, del ejemplo 2 de la sección 2.3.

Aplicando los algoritmos de puntos interiores con esta regla de parada difusa a los mismos problemas utilizados en la sección 2.4 obtenemos los resultados mostrados en las Tablas 2.8 y 2.9. En estos algoritmos, utilizando una función de pertenencia no lineal para definir la regla de detención difusa se pueden emplear grados de tolerancias más realistas, mientras que si se usa una función de pertenencia lineal se tienen que utilizar grados de tolerancia muy altos pero aún así, se obtienen peores aproximaciones. Por otro lado, de entre los tres algoritmos de puntos interiores, el algoritmo primal-dual requiere menor número de iteraciones para alcanzar buenas aproximaciones, aunque el inconveniente radica en la forma de obtener una solución inicial primal y dual simultáneamente.

2.6 EJEMPLOS NUMÉRICOS COMPARATIVOS

Como dijimos al inicio de este capítulo, vamos a ilustrar las ventajas de utilizar el criterio de parada difuso en los algoritmos de PL mediante dos ejemplos. El primer ejemplo es el mismo que se ha utilizado en la sección 2.2 al ilustrar la versión difusa del algoritmo simplex, el segundo, es el ejemplo de Klee-Minty [103] que muestra que el algoritmo simplex es de tiempo de ejecución no polinomial y veremos allí que los algoritmos de puntos interiores son más eficientes y, sobre todo, la ventaja de utilizar el criterio de parada difuso.

2.6.1 Ejemplo 1 :

Se trata de resolver:

$$\max \quad z = 4x_1 + 6x_2 + 8x_3 + 5x_4$$

$$\begin{aligned} s.a. \qquad x_1 + 3x_2 + 2x_3 + 4x_4 &\le 20 \\ 2x_1 + 3x_2 + 6x_3 + 4x_4 &\le 25 \\ x_1 + x_2 &\le 5 \\ x_1 + 2x_2 &\le 8 \\ 4_3 + 3x_4 &\le 12 \\ x_1, x_2, x_3, x_4 &\ge 0. \end{aligned}$$

Este ejemplo, originalmente de tamaño 77, resulta ser de tamaño 135 al transformarlo a la forma estándar para poder aplicar el algoritmo simplex (AS), el algoritmo

de Vanderbei-Miketon-Freedman (AVMF), el algoritmo de Barnes (AB), el algoritmo primal (AP), al algoritmo dual (AD) y el algoritmo primal-dual (APD).

Para aplicar el algoritmo original de Karmarkar (AK), debido a que la función objetivo al convertirla en minimización no es "no negativo", se obtiene un sistema de ecuaciones de tamaño 377 que, después de transformarlo en la forma estándar de Karmarkar resulta un ejemplar de tamaño 21350. Para obtener la solución óptima de este ejemplar mediante el algoritmo de Karmarkar, se requiere un máximo número de 5.124.000 iteraciones aproximadamente.

En la Tabla 2.10 en la fila señalada como "no difusa" se presentan las soluciones obtenidas utilizando los algoritmos con el criterio de parada original. En los dos grupos de líneas siguientes se muestran las soluciones obtenidas con los respectivos algoritmos en los que se han introducido criterios de parada difusos. En la primera línea se utiliza una función de pertenencia lineal y en la segunda, una función no lineal.

En la columna con encabezado α se indica el grado de tolerancia del incumplimiento de la condición de parada; en las demás columnas, para cada algoritmo se consigna el error que se comete al admitir una solución aceptable con respecto a la solución exacta y el número de iteraciones realizadas para obtenerla. Para expresar el error se utiliza una de las fórmulas más usadas en los algoritmos aproximados:

$$error = \frac{solución\ exacta\ -\ solución\ aproximada}{solución\ exacta} \times 100.$$

Se observa claramente que el uso de un criterio de parada difuso reduce el número de iteraciones. Lo más sorprendente es en los algoritmos de Barnes, dual y primal-dual, con criterio de parada difuso con sólo el 20% de iteraciones se obtiene el mismo resultado que sin el criterio de parada difuso, esto indica que para ejemplos particulares existen iteraciones que no mejoran sustancialmente el resultado, por lo tanto sólo consumen tiempo y recursos, situación que puede evitarse usando un criterio de parada difuso.

En general, el uso de un criterio de parada difuso en los algoritmos de PL proporciona una reducción muy ventajosa del número de iteraciones, entre el 10% y el 50%, consiguiendo unos resultados aceptables muy cercanos al óptimo.

α	AS		AK		AVMF		AB		AP		AD		APD	
	error	iteración	error	iteración	error	iteración	error	iteración	error	iteración	error	iteración	error	iteración
NO DIFUSO	0.00	4	0.00	5124000	$5x10^{-6}$	19	0.00	9	$5x10^{-8}$	9786	0.00	12232	0.00	9
DIFUSO-LINEAL														
0.5	1.83	3	$1.8x10^{113}$	17	63.40	1	9.52	0	1.77	2	0.57	2	3.53	2
0.8	1.83	3	$7.3x10^{102}$	25	13.59	2	9.52	0	1.77	2	0.57	2	3.52	2
DIFUSO-NO LINEAL														
0.5	0.00	4	43.72	1406	6.48	3	9.52	0	0.15	4	0.005	5	0.04	5
0.8	0.00	4	0.0238	3054	0.007	9	0.00	6	0.0004	2480	0.00	2062	0.00	8

Tabla 2.10: Soluciones mediante algoritmos con y sin criterio de parada difuso.

Este logro, que constituye la mayor ventaja del uso de un criterio de parada difuso, se aprecia mejor en algoritmos menos eficientes, como es el caso del algoritmo de Karmarkar original con respecto a las modificaciones del mismo, como son los algoritmos de Vanderbei-Meketon-Freedman, Barnes y los tres algoritmos de puntos interiores.

En la Tabla 2.10 al igual que en las Tablas anteriores se observa la ventaja de utilizar una función de pertenencia no lineal y particularmente convexa con respecto a una función de pertenencia lineal, ventaja que se hace más evidente en el algoritmo de Karmarkar.

4.5.2 Ejemplo 2: problema de Klee-Minty:

En 1972 Klee y Minty [103] construyeron un ejemplo de 50 variables y 50 restricciones cuya resolución mediante el algoritmo simplex requería 300,000 años. La forma general de este ejemplo es la siguiente:

$$\max \quad x_1 + x_2 + x_3 + \ldots + x_{n-1} + x_n$$

s.a:

$$\begin{aligned} x_1 &\leq 1 \\ 2x_1 + x_2 &\leq 3 \\ 2x_1 + 2x_2 + x_3 &\leq 9 \\ &\vdots \\ 2x_1 + 2x_2 + 2x_3 + \cdots + 2x_{n-1} + x_n &\leq 3^{n-1} \\ x_1, \cdots, x_n &\geq 0 \end{aligned}$$

Para obtener la solución óptima mediante el algoritmo simplex se requieren 2^n-1 iteraciones. En ejemplos como éste resulta fundamental la introducción del criterio de parada difuso, y ya no sólo en el algoritmo simplex sino también en algoritmos más eficientes como son los algoritmos de puntos interiores.

Ilustramos los resultados de aplicar los algoritmos con criterio de parada difuso estudiados en el ejemplo de Klee-Minty para n=6 (6 variables y 6 restricciones) en la Tabla 2.11. El ejemplo tal como está enunciado (n=6) es de tamaño 99, y expresado en su forma

estándar es de tamaño 147. Si combinamos el ejemplo con su dual y con la condición de igualdad de las funciones objetivo en el caso óptimo, se obtiene un sistema de tamaño 534. Consecuentemente, la forma estándar de Karmarkar alcanza un tamaño 41.464, por lo cual este algoritmo requeriría a lo sumo 12.936.768 de iteraciones para obtener la solución óptima. Ya que es elevado el número de iteraciones que se requieren, no se ha intentado resolver el ejemplo de Klee-Minty mediante el algoritmo de Karmarkar.

Todo lo expresado a la luz de los resultados mostrados en la Tabla 2.10 respecto del primer ejemplo de esta sección se ve corroborado y aún reforzado con los resultados mostrados en la Tabla 2.11. La nota más importante que recogemos de esta última Tabla es el hecho de que el porcentaje del error promedio con respecto a la solución exacta es tan sólo de 5.35, obtenido en tan sólo el 40.1727 % de iteraciones, cuando la función de pertenencia es convexa y el grado de aceptación α=0.8. Proporciones semejantes se pueden obtener también al trabajar con los datos de la Tabla 2.10.

Estos dos ejemplos son pequeñas muestras de las grandes bondades del uso de criterios de parada difusos en los algoritmos denominados exactos. La ventaja de los criterios de parada difusos se perciben mejor cuanto mayores sean las dimensiones de los problemas de PL.

α	AS		AVMF		AB		AP		AD		APD	
	error	iteración	error	iteración	error	iteración	error	iteración	error	iteración	error	iteración
NO DIFUSO	0.00	63	0.00	52	0.00	302	0.00	2584	0.00	2586	0.00	12
DIFUSO-LINEAL												
0.5	91.36	11	0.41	39	25.31	5	14.14	2	18.991	2	8.37	2
0.8	67.08	30	0.41	39	25.31	5	14.14	2	7.076	3	8.37	2
DIFUSO-NO LINEAL												
0.5	88.88	15	0.041	41	24.69	7	0.0014	7	0.186	6	0.058	5
0.8	23.46	44	0.00	46	8.64	13	0.00012	211	0.00029	93	0.00004	8

Tabla 2.11: Soluciones mediante algoritmos con y sin regla de parada difusa.

Capítulo 3

CRITERIOS DE PARADA DIFUSOS EN ALGORITMOS DEL PROBLEMA DE LA MOCHILA Y DEL VIAJANTE DE COMERCIO

3.1 INTRODUCCIÓN

En el *problema de la mochila* se trata de llenar una mochila con todos los objetos posibles que quepan de un total de n objetos, de tal manera que el valor de los objetos transportados sea el mayor posible. Dicho en otras palabras, el objetivo es llenar la mochila de tal manera que se maximice el valor de los objetos transportados respetando la limitación de la capacidad impuesta.

El muy conocido *problema del viajante de comercio*, que brevemente se conoce como TSP (Traveling Salesman Problem o Traveling Salesperson Problem) consiste en determinar el programa de recorrido de una visita a cada una de n ciudades de tal manera que la distancia total sea el menor y que no se repita ninguna de ellas por más de una vez.

En este capítulo exploraremos la aplicabilidad de los criterios de parada difusos a estos dos importantísimos problemas.

3.2 EL PROBLEMA DE LA MOCHILA

El "*problema de la mochila*", cuyo nombre apareció por primera vez en 1957 sugerido por Dantzig [45] para referirse al problema que Bellman [12], unos años antes, denominó como "*problema de la carga*"(loading problem), es un problema simple en su formulación, como se puede observar, pero muy complejo para resolverlo, tal es así que constituye un problema test para medir la eficiencia de diferentes algoritmos.

La importancia de este problema se puede percibir si lo reformulamos en otros términos como por ejemplo, "se dispone de un capital c para financiar proyectos de inversión,

existen n proyectos candidatos que en total requieren capital muy superior al disponible. El problema reside en seleccionar aquellos proyectos de tal manera que los beneficios sean los mayores posibles.

A partir del enunciado anterior y de otros muchos alternativos, resulta evidente la aplicabilidad del problema de la mochila en diversos problemas como los de selección de proyectos y asignación de capital de inversión, los problemas de corte de stock (Cutting Stock), o problemas de los de carga, etc. Un resumen y las correspondientes referencias de estas y otras aplicaciones del problema de la mochila puede encontrarse en [175].

Puesto que el problema de la mochila es NP-completo [134], no admite algoritmos con tiempo de ejecución polinomial en n, aunque sí pseudo polinomiales, es decir, algoritmos cuyos tiempos de ejecución estén acotados por un polinomio en n y c. Sin embargo, existen algoritmos de tipo enumerativo que resuelven el problema de la mochila, como veremos más adelante, en tiempos de ejecución acotados por un polinomio en n, que en el "peor caso" pueden tener un crecimiento exponencial. Estos últimos algoritmos son denominados *algoritmos exactos.*

Otros algoritmos, denominados *aproximados,* determinan soluciones factibles en tiempos de ejecución polinomial en n. Estas soluciones no óptimas (con algunas excepciones, naturalmente) constituyen cotas inferiores de la solución óptima

En el presente trabajo, al igual que en el capítulo 2, flexibilizaremos algunos algoritmos exactos de tal manera que se comporten como algoritmos aproximados. Pero antes, en la sección 3.2.1 describiremos la mayoría de los modelos clasificados dentro del grupo de problemas de la mochila, aunque la flexibilización la haremos para los algoritmos más importantes del problema de la mochila en su caso simple. Analizamos tales algoritmos en la sección 3.2.3.

El comportamiento de los algoritmos aproximados para un ejemplar específico se mide comparando con la solución óptima, o bien si se desconoce la solución óptima, con alguna cota superior. Puesto que nuestro trabajo consistirá en intentar hacer que los algoritmos exactos se comporten como los algoritmos aproximados, resulta importante hacer las comparaciones respectivas. Teniendo en cuenta que no es muy fácil obtener una solución exacta para el problema de la mochila de grandes dimensiones, optamos por las comparaciones con alguna cota. Por ello en la sección 3.2.2 revisaremos algunos algoritmos

importantes para calcular las cotas superiores del problema de la mochila. En la 3.2.4 se exponen las modificaciones que proponemos, la introducción de criterios de parada difusos en los algoritmos que resuelven el problema de la mochila, y particularmente ilustramos dichos criterios difusos en dos algoritmos de ramificación y acotamiento, y en dos algoritmos de programación dinámica que estudiaremos en la sección 3.2.3. Finalmente en la sección 3.2.5 presentamos los resultados de experimentos numéricos, las comparaciones y las conclusiones respectivas.

3.2.1 Formulación y tipos de problemas de la mochila

El problema de la mochila formulado literalmente en la sección anterior puede ser matemáticamente expresado enumerando los objetos de 1 a n que de aquí en adelante llamaremos *ítems*, introduciendo números w_j, p_j (j=1,...,n) llamados *peso* y *valor* respectivamente del j-ésimo ítem e introduciendo variables binarias x_j (j=1,...,n) con el siguiente significado:

$$x_j = \begin{cases} 1 & \text{si el item } j \text{ es seleccionado} \\ 0 & \text{en otro caso} \end{cases}$$

Por lo tanto el problema de la mochila consiste en seleccionar aquellos objetos (ítems) de entre los n de tal manera que la suma total de sus valores respectivos sea el mayor posible, pero que la suma de sus pesos respectivos no sea superior a la capacidad c de la mochila, es decir, se trataría de resolver el siguiente problema:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ s.a. \quad & \sum_{j=1}^{n} w_j x_j \le c \\ & x_j \in \{0,1\}, \quad j=1,2,...,n. \end{aligned} \tag{3.1}$$

Este problema prototipo que en adelante denominaremos problema clásico de la mochila o simplemente problema de la mochila (PM), es conocido como el problema de la mochila 0-1 (0-1 Knapsack Problem).

En la generalidad de los casos no se imponen condiciones sobre los parámetros del modelo (3.1). Debido a que casi la totalidad de los algoritmos construidos consideran que

todos los parámetros son enteros positivos, si no se especifica lo contrario, también consideraremos que todos los parámetros son enteros positivos. Cada ítem es de menor peso que la capacidad y la suma de los pesos de los n ítems es superior a la capacidad de la mochila, y sobre todo, consideraremos que los ítems están ordenados, en orden decrecientes del valor por unidad de peso, es decir:

$$\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \ldots \geq \frac{p_n}{w_n}. \tag{3.2}$$

Cualquier situación diferente se puede reducir siempre a este caso. Así por ejemplo, si algunos parámetros son fracciones, multiplicamos por un factor apropiado. Si algún ítem tiene valor negativo simplemente se descarta, y si tiene peso negativo pero con valor positivo se selecciona; luego, se construye un nuevo problema excluyendo los ítems descartados y admitidos inicialmente; y si la suma de los pesos es menor que la capacidad, la solución es trivial pues todos los ítems son seleccionados para meter a la mochila.

Aunque en este trabajo fijaremos nuestra atención sólo en el problema clásico de la mochila, no podemos dejar de mencionar los otros tipos de problemas de la mochila derivados del problema clásico, también de importancia, que darían lugar a un posterior trabajo ya que su estudio en profundidad desbordaría los límites de esta memoria. En cualquier caso pueden destacarse los siguientes:

a) Cuando los n ítems se agrupan en r grupos N_k (k=1,...,r) y se impone la condición de que debe elegirse necesariamente un ítem de cada subgrupo, se obtiene el denominado *problema de la mochila de múltiple elección (multiple-choice Knapsack problem)*, que se formula de la siguiente manera:

$$\max \sum_{j=1}^{n} p_j x_j$$

$$s.a. \sum_{j=1}^{n} w_j x_j \leq c$$

$$\sum_{j \in N_k} x_j = 1, \quad k=1,\ldots,r$$

$$x_j \in \{0,1\}, \quad j=1,2,\ldots,n$$

$$\{1,2,3.....,n\} = \bigcup_{k=1}^{r} N_k.$$

b) Cuando en el PM se admite la posibilidad de elegir a lo sumo b_j unidades del ítem j, el problema se denomina *problema acotado de la mochila* (bounded Knapsack problem):

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ s.a. \quad & \sum_{j=1}^{n} w_j x_j \le c \\ & 0 \le x_j \le b_j, \quad j=1,\ldots,n \\ & x_j \text{ entero}, \quad j=1,2,\ldots,n. \end{aligned}$$

y si $b_j=+\infty$ (j=1,...,n), se denomina *problema no acotado de la mochila* (unbounded Knapsack problem).

c) Un caso especial del PM se presenta cuando $p_j=w_j$ (j=1,2,..,n), lo que se conoce como el *problema de subconjunto suma* (subset-sum problem):

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} w_j x_j \\ s.a. \quad & \sum_{j=1}^{n} w_j x_j \le c \\ & x_j \in \{0,1\}, \quad j=1,2,\ldots,n. \end{aligned}$$

d) Si en el problema acotado de la mochila $p_j=1$ (j=1,...,n) o bien todos los ítems tienen el mismo valor, se obtiene el denominado *problema de hacer cambios* (change-making problem):

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} x_j \\ s.a. \quad & \sum_{j=1}^{n} w_j x_j \le c \\ & 0 \le x_j \le b_j, \quad j=1,\ldots,n \\ & x_j \text{ entero}, \quad j=1,2,\ldots,n. \end{aligned}$$

Naturalmente, se obtiene el correspondiente problema no acotado cuando $b_j=+\infty$ (j=1,...,n).

e) Si en lugar de tener una mochila se tienen m mochilas y además se impone la condición de introducir un ítem a lo sumo en una mochila, el problema se denomina el *problema múltiple de la mochila.* Para la formulación matemática introducimos constantes c_i (i=1,...,m) para expresar las respectivas capacidades de las mochilas, y variables binarias x_{ij} con valor 1, para indicar que el ítem j se ha seleccionado para llenar a la mochila i, y

valor 0 en otros casos; de este modo, tenemos:

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_j$$

$$s.a. \quad \sum_{j=1}^{n} w_j x_{ij} \le c_i \qquad i=1,\ldots,m$$

$$\sum_{i=1}^{m} x_{ij} \le 1, \qquad j=1,\ldots,n$$

$$x_{ij} \in \{0,1\}, \quad j=1,\ldots,n.$$

Cuando en este problema tanto el valor como el peso del ítem varía de acuerdo a la mochila, estamos ante el problema de asignación generalizado.

A su vez, en cada uno de los casos anteriores, puede existir más de una función objetivo. De ser así se originarían los correspondientes *problemas de la mochila multiobjetivo.*

3.2.2 Cotas superiores del problema de la mochila

Sea z(PM) la función objetivo del problema de la mochila. Una cota superior de PM es un número U tal que z(PM)≤U. Para calcular una cota superior se han desarrollado diferentes algoritmos, siendo de mayor importancia los derivados de la relajación continua por una parte y por otra parte los de enumeración parcial. Revisaremos en esta sección las cotas superiores obtenidas mediante la relajación y por enumeración parcial; todas ellas tomadas de [134].

3.2.2.1 Cotas obtenidas mediante relajación

Existen dos tipos importantes de relajación: continua, llamada también relajación en programación lineal, y lagrangiana. Algunos conceptos importantes asociados a cada una de ellas son las siguientes:

3.2.2.1.1 Relajación continua

La relajación continua es una relajación natural y quizás por ello, fue la primera en ser propuesta. Dicha relajación se ha denominado *problema continuo de la mochila,* que denotaremos por C(PM), y se obtiene a partir de (3.1) reemplazando la condición de variable binaria por variable con valores en el intervalo [0,1], es decir:

$$\text{C(PM):} \quad \begin{aligned} &\max \sum_{j=1}^{n} p_j x_j \\ &s.a. \sum_{j=1}^{n} w_j x_j \le c \\ &0 \le x_j \le 1, \quad j=1,2,\ldots,n. \end{aligned}$$

Esta relajación no es sino el mismo problema inicial pero en el que a la hora de seleccionar los objetos, se puede proceder a fraccionarlos cuando ya no es posible introducir los objetos enteros. Bajo esta condición, el procedimiento de selección es muy simple, ya que se seleccionarán todos los objetos que quepan en la mochila hasta alcanzar el (s-1)-ésimo ítem de tal manera que el siguiente ítem s ya no quepa; luego, este último objeto se romperá en las fracciones necesarias hasta llenar completamente la mochila. Este procedimiento, propuesto por Dantzig en 1957, fue utilizado durante dos décadas como el mejor método para resolver el problema de la mochila.

Ítem crítico: El ítem s mencionado en el párrafo anterior se denomina ítem crítico y formalmente lo definimos como:

$$s = \min \left\{ j: \sum_{i=1}^{j} w_i > c \right\} \tag{3.3}$$

Solución continua: Luego la solución óptima x* del problema C(PM) es:

$$\begin{aligned} x_j^* &= 1 \quad \text{para} \quad j = 1,\ldots,s-1 \\ x_j^* &= 0 \quad \text{para} \quad j = s+1,\ldots,n \\ x_s^* &= \frac{\bar{c}}{w_s} \end{aligned}$$

donde

$$\overline{c} = c - \sum_{j=1}^{s-1} w_j \tag{3.4}$$

y el valor de la solución óptima de C(PM) es:

$$z(\mathrm{C(PM)}) = \sum_{j=1}^{s-1} p_j + \overline{c}\,\frac{p_s}{w_s}. \tag{3.5}$$

La cota de Dantzig: Teniendo en cuenta la condición de variable entera para el problema de la mochila inicial se obtiene una cota superior denominada cota de Dantzig:

$$U_1 = \lfloor z(\mathrm{C(PM)}) \rfloor = \sum_{j=1}^{s-1} p_j + \left\lfloor \overline{c}\,\frac{p_j}{w_j} \right\rfloor, \tag{3.6}$$

donde $\lfloor a \rfloor$ nota el mayor de los números enteros no mayores que a.

Martello y Toth obtienen una cota superior mejor que la de Dantzig imponiendo la condición de que la variable crítica x_s sea entera. Así, se tienen:

$$U^0 = \sum_{j=1}^{s-1} p_j + \left\lfloor \overline{c}\,\frac{p_{s+1}}{w_{s+1}} \right\rfloor \tag{3.7}$$

$$U^1 = \sum_{j=1}^{s-1} p_j + \left\lfloor p_s - (w_s - \overline{c})\,\frac{p_{s-1}}{w_{s-1}} \right\rfloor \tag{3.8}$$

donde s y $\overline{c}$ son valores definidos por (3.3) y (3.4) respectivamente.

***Cota superior de Martello-Toth*:** Luego una cota superior mejor es:

$$U_2 = \max\,(U^0, U^1); \tag{3.9}$$

Otra mejora de la cota superior se obtiene a raíz de la propuesta de Hudson para mejorar (3.7), por una parte, y por otra, de la propuesta de Fayard y Plateau quienes realizan cálculos que mejoran (3.8), del siguiente modo:

$$\overline{U}^0 = \sum_{\substack{j=1 \\ j\neq s}}^{\sigma^0(s)-1} p_j + \left\lfloor \left(c - \sum_{\substack{j=1 \\ j\neq s}}^{\sigma^0(s)-1} w_j \right) \frac{p_{\sigma^0(s)}}{w_{\sigma^0(s)}} \right\rfloor \tag{3.10}$$

$$\overline{U}^1 = p_s + \sum_{j=1}^{\sigma^1(s)-1} p_j + \left\lfloor \left(c - w_s - \sum_{j=1}^{\sigma^1(s)-1} w_j \right) \frac{p_{\sigma^1(s)}}{w_{\sigma^1(s)}} \right\rfloor \tag{3.11}$$

donde

$$\sigma^0(j) = min \left\{ k : \sum_{\substack{i=1 \\ i \neq j}}^{k} w_i > c \right\} \tag{3.12}$$

y

$$\sigma^1(j) = min \left\{ k : \sum_{i=1}^{k} w_i > c - w_j \right\} \tag{3.13}$$

como consecuencia se obtiene una cota superior mejor, que es la conocida como ***cota superior de Hudson-Fayard-Plateau***:

$$U_{\square} = \max(\bar{U}^{\bar{x}}, \bar{U}^{\square}) \tag{3.14}$$

Estas cotas satisfacen la relación $U_3 \leq U_2 \leq U_1$, de donde se refleja la condición de que U_3 es una cota superior mejor que U_2, y ésta a su vez mejor que U_1.

3.2.2.1.2 Relajación lagrangiana

La otra relajación que hemos mencionado es la lagrangiana, que notaremos por L(PM,λ), donde λ es un multiplicador no negativo, y que constituye el problema:

$$\max \sum_{j=1}^{n} p_j x_j + \lambda \left(c - \sum_{j=1}^{n} w_j x_j \right)$$
$$\text{s.a.} \quad x_j \in \{0,1\} \quad j = 1, \ldots, n.$$

cuya función objetivo se simplifica en:

$$z(\text{L(PM},\lambda)) = \sum_{j=1}^{n} \tilde{p}_j x_j + \lambda c \tag{3.15}$$

con la notación $\tilde{p}_j = p_j - \lambda w_j, \quad j = 1, \ldots, n.$

La solución óptima de la relajación lagrangiana se obtiene fácilmente haciendo:

$$\tilde{x}_j = \begin{cases} 1 & \text{si } \tilde{p}_j > 0 \\ 0 & \text{si } \tilde{p}_j < 0 \end{cases} \qquad (3.16)$$

y luego, definiendo

$$J(\lambda) = \{ j: p_j/w_j > \lambda \},$$

con los cuales el valor de la solución resulta:

$$z(\mathrm{L(PM}, \lambda)) = \sum_{j \in J(\lambda)} \tilde{p}_j + \lambda c \,.$$

Si $\lambda^* = p_s / w_s = \text{mín}\{\lambda_j, j \in J(\lambda)\}$,donde s es el ítem crítico, se cumple $z(\mathrm{C(PM)})=z(\mathrm{L(PM},\lambda^*))$, y $\tilde{p}_j$ viene a ser:

$$p_j^* = p_j - w_j \frac{p_s}{w_s} \qquad (3.17)$$

Utilizando este valor, Müller-Merbach propuso la llamada ***cota de Müller-Merbach***:

$$U_4 = max \left(\sum_{j=1}^{s-1} p_j, max \left\{ \left\lfloor z(C(PM)) - \left| p_j^* \right| \right\rfloor : j = 1,..., n, j \neq s \right\} \right) \qquad (3.18)$$

Esta cota es mejor que la cota de Dantzig, pero, en general, no es mejor que las cotas U_2 y U_3, aunque en algunas situaciones se cumple que $U_4 < U_3 \leq U_2 \leq U_1$.

Partiendo de las ideas usadas en el cálculo de las cotas anteriores, Dudzinski y Walukievicz obtuvieron una cota mejor que las anteriores. Para ello, primero, usando la solución óptima x_j^* de la relajación continua, definen una solución factible $\hat{x}$ de la siguiente manera:

$$\hat{x}_j = \begin{cases} x_j^* & j \neq s \\ 0 & j = s \end{cases}$$

luego para cada k tal que $\hat{x}_k = 0$:

$$\text{si} \quad w_k \leq c - \textstyle\sum_{j=1}^{n} w_j \hat{x}_j \quad \text{hacen} \quad \hat{x}_k = 1.$$

Ahora sean

$$\hat{N} = \{ j \in \{1,\ldots,n\} \setminus \{s\} : \hat{x}_j = 0 \}$$

$$\hat{U}^0 = min\left(\overline{U}^0, max\left\{ \left\lfloor z(C(PM)) + p_j^* \right\rfloor : j \in \hat{N} \right\} \right)$$

$$\hat{U}^1 = min\left(\overline{U}^1, max\left\{ \left\lfloor z(C(PM)) - p_j^* \right\rfloor : j = 1,\ldots,s-1 \right\} \right)$$

donde $\overline{U}^0$ y $\overline{U}^1$ son dados en (3.10) y (3.11). Luego la ***cota de Dudzinski-Walukiewicz*** está dada por:

$$U_5 = max\left\{ \hat{U}^0, \hat{U}^1, \sum_{j=1}^{n} p_j \hat{x}_j \right\} \tag{3.19}$$

que es la mejor de las cuatro cotas definidas anteriormente.

3.2.2.2 Cota obtenida mediante enumeración parcial

Teniendo como referencia el ítem crítico s, Martello y Toth [133] propusieron otra forma de determinar una cota superior del problema de la mochila. Para ello seleccionan números r y t tales que $1 < r \le s$ y $s \le t < n$, y construyen una solución factible de PM haciendo $x_j=1$ para $j<r$, $x_j=0$ para $j>t$ y hallando la solución óptima del sub-problema PM(r,t) definido por los ítems r,r+1,...,t con capacidad

$$c(r) = c - \sum_{j=1}^{r-1} w_j.$$

La solución óptima del subproblema PM(r,t) se calcula mediante el árbol de decisión binaria para $j=r,r+1,\ldots,t$, generando pares de nodos de decisión haciendo $x_j = 1$ y $x_j = 0$ en cada nodo k. Para cada nodo k del árbol resultante, sea f(k) el ítem que ha generado el nodo k (haciendo $x_{f(k)} = 1$ o $x_{f(k)} = 0$). El conjunto de los nodos terminales (ramas) del árbol puede ser particionado en:

$$L_1 = \left\{ l : \sum_{j=r}^{f(l)} w_j x_j^l > c(r) \right\} \qquad \text{(Ramas infactibles)}$$

$$L_2 = \left\{ l : f(l) = t \text{ y } \sum_{j=r}^{f(l)} w_j x_j^l \le c(r) \right\} \qquad \text{(Ramas factibles)}$$

Luego para cada $l \in L_1 \cup L_2$, sea u_l una cota superior de PM, definida de la siguiente manera:

$$u_l = \begin{cases} \left\lfloor p^l - d^l \dfrac{p_{r-1}}{w_{r-1}} \right\rfloor & si \quad l \in L_1 \\ \left\lfloor p^l + d^l \dfrac{p_{t+1}}{w_{t+1}} \right\rfloor & si \quad l \in L_2 \end{cases} \qquad (3.20)$$

donde:

$$p^l = \sum_{j=1}^{r-1} p_j + \sum_{j=r}^{f(l)} p_j x_j^l \qquad y \qquad d^l = |c(r) - \sum_{j=r}^{f(l)} w_j x_j^l|$$

y el máximo de estas cotas es la denominada ***Cota de Martello y Toth:***

$$U_6 = \max \{ u_l : l \in L_1 \cup L_2 \}. \qquad (3.21)$$

Esta cota es mejor que las cotas anteriores excepto la cota de Dudzinsky y Walukiewicz. Por tanto, la mejor cota de entre las seis cotas dadas es:

$$U = \min (U_5, U_6).$$

3.2.3 Algoritmos de solución del problema de la mochila

Para cada uno de los modelos presentados en la sección 3.2.1 y para otros casos especiales existen muchos algoritmos de solución. Una muy buena recopilación de dichos algoritmos puede encontrarse en [134]; también [175] recoge un buen número de algoritmos para resolver particularmente el problema de la mochila en su caso más simple. Los algoritmos que presentamos aquí se han recogido de estos dos libros.

Como se sabe, dentro de la programación entera y en particular para problemas 0-1 los métodos de *ramificación y acotación* y el de la *programación dinámica* son los más utilizados. Puesto que el PM es uno de los casos de la programación binaria, no es una excepción en dicha utilización, por ello y porque basándose en estos métodos se han

construido los algoritmos más eficientes para resolver el PM, en este trabajo hemos elegido dos algoritmos basados en el método de ramificación y acotación que veremos en la sección 3.2.3.2 y dos algoritmos basados en la programación dinámica que veremos en la sección 3.2.3.3. Además, previamente en la sección 3.2.3.1, revisamos un algoritmo aproximado y una heurística de tipo voraz (greedy). Sobre cada uno de los algoritmos que estudiamos, explicamos su metodología general y presentamos su procedimiento detallado en la forma de pseudo-código-pascal, siguiendo la misma forma que utilizan Martello y Toth [134].

3.2.3.1 Algoritmos de aproximación

Consideramos dentro de esta sección aquellos algoritmos que resuelven el PM en un tiempo de ejecución polinomial, pero que no garantizan la optimalidad de la solución obtenida; razón por la cual el valor correspondiente a la solución obtenida por un algoritmo de aproximación constituye solamente una cota inferior del valor óptimo.

Existen diferentes tipos de algoritmos aproximados, los llamados heurísticas o los propiamente algoritmos aproximados. Una revisión de los más importantes algoritmos de este tipo poden encontrarse en [134].

Puesto que nuestro objetivo no es hacer una revisión de los algoritmos aproximados, sino que sólo utilizarlos como una herramienta, en este trabajo hemos elegido un algoritmo voraz por ser el más popular y eficiente de los algoritmos heurísticos para resolver PM y el algoritmo aproximado de Sahni por ser uno de los más simples de este tipo.

3.2.3.1.1 El algoritmo de tipo voraz

Los algoritmos voraces son los más fáciles de entender e implementar y típicamente se utilizan para resolver problemas de optimización. Se puede decir que su procedimiento se asemeja al de un "método de la fuerza bruta", característica por la cual actúan de modo inmediato basándose sólo en la información actual, y sin tener en cuenta los efectos futuros, y sobre todo, no revisan (sino que olvidan) las decisiones tomadas anteriormente.

Los algoritmos voraces, pese a su simplicidad, aparecieron tardíamente -en 1971- introducidos por Edmonds [59]. Por su tosquedad, son inaplicables en muchos casos, y

pueden proporcionar resultados erróneos, haciendo creer por ejemplo, que un problema no tiene solución aún teniéndola. Generalmente las soluciones que proporciona no son óptimas. A pesar de todas sus limitaciones en muchas circunstancias son algoritmos muy eficientes. De ahí que sean frecuentemente utilizados como heurísticas en problemas NP-completos. En este trabajo los utilizaremos como referencia en la definición de los criterios de parada difusos.

Los algoritmos voraces aplicados al problema de la mochila dan lugar al algoritmo que aquí llamaremos *algoritmo voraz*, cuya metodología de funcionamiento se basa en el más elemental sentido común. Pués, sabiendo que los ítems están ordenados de mayor a menor valor por unidad de peso, este algoritmo los introduce ordenadamente, desde el primero hasta el último, excluyendo sólo el ítem que no quepa en el espacio restante de la mochila.

El algoritmo voraz, puede también entenderse como la modificación de la solución continua (solución de la relajación de PM en programación lineal vista en 3.2.2.1.1). Esta modificación consiste en asignar valor cero a la única variable con valor fraccionario; luego se asigna valor uno (si existe) a aquella variable que tiene valor cero y que corresponde a un ítem cuyo peso no excede al espacio restante de la mochila; y así sucesivamente hasta llenar la mochila o hasta terminar de verificar los ítems restantes. Basándonos en esto, el algoritmo voraz para resolver el problema de la mochila, ordenado como en (3.2), podemos resumirlo de la siguiente manera:

Etapa Inicial:

Determinar el ítem crítico s; luego, hacer:

$$x_j = 1, \quad j = 1,\ldots,s-1,$$
$$\bar{c}_{s-1} = c - \sum_{j=1}^{s-1} w_j,$$
$$z_{s-1} = \sum_{j=1}^{s-1} p_j$$

Etapa Iterativa:

Para $j = s, s+1,\ldots,n$:

$$x_j = \begin{cases} 1 & \text{si } w_j \le \bar{c}_{j-1} \\ 0 & \text{si } w_j > \bar{c}_{j-1} \end{cases},$$
$$\bar{c}_j = \bar{c}_{j-1} - w_j x_j,$$
$$z_j = z_{j-1} + p_j x_j.$$

Naturalmente, la regla de parada implícita es $j=n$, por tanto el valor final de la función

objetivo es $z^*=z_n$.

El procedimiento detallado del algoritmo voraz que acabamos de resumir podemos encontrarlo en el Cuadro 3.1.

Procedimiento **Voraz**:

```
Input: n, c, (p_j), (w_j);
output: z^g, (x_j);
begin
    c̄ := c;
    z^g := 0;
    j* := 1;
    for j:=1 to n do
        begin
            if w_j > c then x_j:=0
            else
                begin
                    x_j := 1;
                    c̄ := c̄ - w_j;
                    z^g := z^g + p_j;
                end;
            if p_j > p_j* then j*:=j
        end;
    if p_j* > z^g then
        begin
            z^g := p_j*;
            for j:=1 to n do x_j:=0;
            x_j*:=1;
        end;
end;
```

Cuadro 3.1: El Procedimiento del algoritmo Voraz en elPM.

3.2.3.1.2 Algoritmo aproximado de Sahni

El algoritmo de Sahni [164] es similar al procedimiento del algoritmo voraz. La idea de este algoritmo es llenar la capacidad restante de la mochila una vez introducido un conjunto M de k ítems. Para ello, en primer lugar se generan todos los subconjuntos M posibles de k items para introducirlos en la mochila, posteriormente rellena el espacio restante. El procedimiento de este algoritmo se presenta aquí en dos procesos anidados: S(k) y GS.

En el proceso S(k) (k es un entero no negativo) del Cuadro 3.2 se generan todos los subconjuntos posibles M de k ítems para introducirlos en la mochila, y mediante el proceso GS del Cuadro 3.3 se rellena el espacio restante de la mochila. Al final de cada iteración, es decir, para cada conjunto M se obtiene su valor. Si el valor obtenido es mejor se reserva.

El algoritmo finaliza cuando se haya terminado de analizar todos los subconjuntos con M con k elementos, la solución corresponde al valor último reservado.

Puesto que a mayor valor k el número de conjuntos M es muy grande, el algoritmo de Sahni se utiliza generalmente sólo para valores pequeños de k (k=1,2,3).

Procedimiento **S(k)**

```
input: n, c, (p_i), (w_i);
output; z^h, X^h;
begin
    z^h := 0;
    for each M ⊂ {1,2,...,n}: |M| ≤ k, Σ_{j∈M} w_j ≤ c do
        begin
            call GS;
            if z^g + Σ_{j∈M} p_j > z^h then
                begin
                    z^h := z^g + Σ_{j∈M} p_j;
                    X^h := X ∪ M;
                end;
        end;
end;
```

Cuadro 3.2. Procedimiento del algoritmo de Sahni.

Procedimiento **GS**

input: n, c, (p_i), (w_i), M;
output; z^g, X;
begin

$z^g := 0$;
$\hat{c} := c - \sum_{j \in M} w_j$;
$X := \Phi$;
for $j := 1$ **to** n **do**
if $j \notin M$ *and* $w_j \leq \hat{c}$ **then**
begin
$z^g := z^g + p_j$;
$\hat{c} := \hat{c} - w_j$;
$X := X \cup \{j\}$;

end;

Cuadro 3.3. Procedimiento en el proceso GS del algoritmo aproximado de Sahni.

3.2.3.2 Algoritmos de ramificación y acotación en PM

Las primeras ideas del método de método de *ramificación y acotación* (branch-and-bound), y se sustituye en lo que sigue por RA, aparecieron en 1954 en el trabajo de Dantzig, Fulkerson y Johnson [48], pero las ideas centrales fueron propuestas por Land y Doig en 1960 como un método de enumeración; y la expresión "ramificación y acotación" fue acuñado por Little, Murty, Sweeney y Karel [120] en 1963, mientras que como *método de ramificación y acotación* para resolver problemas generales de optimización combinatorial fue introducido por Balinski] [7] en 1965.

En la formulación de un problema de PE de cualquier situación del mundo real se puede asumir que cada variable entera está acotada; por tanto, cualquier problema con un número finito de variables tendrá un número finito de soluciones factibles. El método de RA enumera inteligentemente todos estos puntos factibles. La enumeración inteligente consiste en realizar una prueba de optimalidad de la solución actual basándose en los procesos de *ramificación y acotación* con la finalidad de no enumerar todos los puntos factibles para encontrar una solución óptima. La denominación de *ramificación* se refiere al proceso de

partición del conjunto factible en subconjuntos, mientras que el término *acotación* se refiere a la determinación de cotas para el valor de la función objetivo sobre cada uno de los subconjuntos obtenidos en la partición.

Dicho en otros términos, el método de RA llamado también como el método de enumeración implícita, resuelve un problema de optimización discreta subdividiendo su conjunto factible sucesivamente en subconjuntos más pequeños, calculando la cota de la función objetivo en cada subconjunto, y usando dichas cotas para descartar ciertos subconjuntos de toda consideración futura. Frecuentemente, las cotas son obtenidas reemplazando el problema sobre un subconjunto dado por un problema mucho más fácil denominado problema relajado. El procedimiento termina cuando para cada subconjunto obtenido en las particiones se ha obtenido la solución factible o cuando se ha verificado que no contiene una solución mejor que la obtenida anteriormente. Por tanto, dado un problema de optimización discreta P con $v(P)$ que nota el valor de la solución óptima, al resolver mediante el método de RA se presentan cuatro componentes generales:

i) Una relajación $R_i(P_i)$ de cada sub-problema P_i, es decir, un problema R_i de la misma forma que P_i pero que engloba al sub-problema P_i y sobre todo más fácil de resolver.

ii) Una regla de ramificación o separación, es decir, una regla que divide el conjunto factible del sub-problema P_i .

iii) Un procedimiento de acotación, es decir, un procedimiento para encontrar $v(R_i)$ para la relajación R_i de cada sub-problema P_i , y

iv) Una regla de selección, es decir, una regla para elegir el siguiente sub-problema P_i que debe ser analizado.

Una presentación formal del método de RA podemos encontrarla por ejemplo en [4] y [136] entre otros.

El primer algoritmo basado en el método de RA desarrollado para aproximar a una solución exacta de un PM fue propuesto por Koleser en 1967. Posteriormente Greenberg y Hegerich propusieron un algoritmo más eficiente que el de Koleser. Basándose en los anteriores algoritmos y siguiendo la estrategia de un algoritmo voraz, Horowitz y Sahni propusieron uno de los algoritmos más eficientes para resolver el PM y sobre todo de fácil implementación. A partir de los anteriores algoritmos y en especial del algoritmo de

Horowitz-Sahni se han derivado muchos algoritmos. Abundante referencias al respecto podemos encontrar en [134], siendo de mayor importancia el algoritmo de Martello y Toth [132] considerado de alta efectividad.

El algoritmo Horowitz-Sahni (HS1), sigue la estrategia de ramificación binaria en profundidad, y consiste en que:

a) En cada nodo se selecciona el ítem *j* con el máximo valor por unidad de peso de entre los ítems que aún no están seleccionados y se generan dos nodos descendientes asignando a x_j respectivamente el valor 1 y 0.

b) La búsqueda continúa desde el nodo asociado con la selección del ítem j (condición x_j=1), es decir, siguiendo la estrategia del algoritmo voraz.

c) Revisa las decisiones tomadas anteriormente mediante movimientos de retroceso.

Cada una de las acciones indicadas arriba, las ejecuta como pasamos a relatar. Un movimiento hacia adelante consiste en insertar el mayor número posible de nuevos ítems consecutivos en la solución actual, y un movimiento de retroceso consiste en borrar el más reciente ítem insertado de la solución actual. Cuando el movimiento hacia adelante es exhaustivo, se calcula la cota superior U_1 correspondiente a la solución actual para comparar con la mejor solución obtenida. Se efectúa un movimiento hacia adelante si se obtiene una mejora, y se retrocede en caso contrario. Cuando se ha analizado el último ítem se ha completado la solución actual y entonces, se actualiza la mejor solución. El ***Criterio de parada***, que es lo que nos interesa, indica que el procedimiento de este algoritmo finaliza cuando no se puede efectuar ningún retroceso.

En el algoritmo HS, escrito detalladamente como se muestra en el Cuadro 3.4 en donde, al igual que en todos los algoritmos que veremos, se considera que los valores y pesos están ordenados de acuerdo a (3.2), y se utilizan las notaciones siguientes:

Procedimiento HS:

input: n, c, (p_j), (w_j);
output: z, (x_j);
begin
1.[Inicialización]

$z:=0$; $\hat{z}:=0$; $\hat{c}:=c$; $p_{n+1}:=0$; $w_{n+1}:=+\infty$; $j:=1$;

2. [Calcular la cota superior U_1]

hallar $r = min\left\{i : \sum_{k=j}^{i} w_k > \hat{c}\right\}$;

$u := \sum_{k=j}^{r-1} p_k + \left\lfloor \left(\hat{c} - \sum_{k=j}^{r-1} w_k\right) p_r / w_r \right\rfloor$;

if $z \geq \hat{z} + u$ **then go to 5 :**

3. [Efectuar un paso hacia adelante]

```
while   w_j ≤ ĉ  do
        begin
              ĉ := ĉ − w_j ;      ẑ := ẑ + p_j ;      x̂ := 1;
              j := j + 1;
        end;

if  j ≤ n  then
        begin
              x̂ := 0;
              j: = j + 1;
        end ;
if  j < n   then  go to 2;
if  j = n   then  go to 3;
```

4. [Actualiza la mejor solución]

```
if   ẑ > z then
        begin
              z: = ẑ;
              for k: = 1 to n do  x_k : = x̂_k ;
        end;
j := n;
if  x_n  = 1  then
        begin
              ĉ := ĉ + w_n ;
              ẑ := ẑ − p_n ;
              x̂_n  := 0;
        end;
```

5. [Retrocede]

```
hallar    i =  max  {k < j: x̂_k  = 1};
if no existe   tal i  then  return;
ĉ: = ĉ + w_i ;
ẑ: = ẑ-p_i ;
x̂_i : = 0;
j: = i + 1;
go to 2;
end;
```

Cuadro 3.4. Procedimiento del algoritmo Horowitz-Sahni.

$(\hat{x}_j)$: representa a la solución actual;

$\hat{z} = \sum_{j=1}^{n} p_j \hat{x}_j$: representa al valor de la solución actual;

$\hat{c} = c - \sum_{j=1}^{n} w_j \hat{x}_j$: representa la capacidad residual;

(x_j) : representa a la mejor solución actual: y

$z = \sum_{j=1}^{n} p_j x_j$: representa al valor de la mejor solución actual.

Por otra parte **el algoritmo de Martello y Toth (MT1)** es una eficiente modificación del algoritmo de HS, realizada por Martelo y Toth [132], en los siguientes:

a) Utiliza la cota superior U_2 en lugar de U_1.

b) El movimiento asociado con la selección del j-ésimo ítem se efectúa en dos fases: i) Construcción de una nueva solución, y ii) Almacenamiento de la solución actual.

En la primera fase, si *j* es el ítem actual, se determina el conjunto N_j con el mayor número de ítems consecutivos a partir del ítem *j* que pueden ser insertados en la mochila, y también se determina la cota superior U_2 correspondiente. Si esta cota superior es menor o igual que el valor de la mejor solución obtenida se hace un movimiento de retroceso; por el contrario, si es mayor se pasa a la siguiente fase. En la segunda fase, existen dos posibilidades, una, que se haya obtenido la solución máxima en cuyo caso se actualiza la mejor solución y no la solución actual para poder efectuar el retroceso en N_j, y la otra, el caso contrario, en que sí se efectúa la inserción de N_j en la solución actual.

c) La tercera modificación consiste en que se realiza un movimiento especial hacia adelante, sobre la base del criterio de dominancia, siempre que después de un movimiento hacia atrás sobre el ítem *i*, la capacidad residual no permita la inserción en la solución actual de ningún ítem siguiente al *i*-ésimo. Esto se debe a que la solución actual puede ser mejorada sólo si el ítem *i* es reemplazado por un ítem con el mayor valor y un peso suficientemente pequeño, o cuando menos, mediante dos ítems que tengan peso global no mayor que el peso *i* más la capacidad residual.

Procedimiento **MT**:

input: n, c, (p_j), (w_j);
output: z, (x_j);
begin

1.[Inicialización]

$z := 0;$ $\hat{z} := 0;$ $\hat{c} := c;$ $p_{n+1} := 0;$ $w_{n+1} := +\infty$
for $k := 1$ **to** n **do** $\hat{x}_k = 0;$
Calcule la cota superior $U = U_2$.
$\overline{w}_1 := 0;$ $\overline{p}_1 := 0;$ $\overline{r}_1 := 0;$ $\tilde{r} := n;$
for $k := n$ **to** 1 **step**-1 **do calcular** $m_k = min\{w_i : i < k\};$
$j := 1;$

2. [Construir una nueva solución]

while $w_j > \hat{c}$ **do**
if $z \geq \hat{z} + \lfloor \hat{c} p_{j+1} / w_{j+1} \rfloor$ **then go to 5 el** $j := j + 1;$
hallar $r = min \left\{ i : \overline{w}_j + \sum_{k=\overline{r}_j}^{i} w_k > \hat{c} \right\};$
$p' := \overline{p}_j + \sum_{k=\overline{r}_j}^{r-1} p_k;$
$w' := \overline{w}_j + \sum_{k=\overline{r}_j}^{r-1} w_k;$
if $r \leq n$ **then** $u := max\left(\lfloor (\hat{c} - w') p_{r+1} / w_{r+1} \rfloor, \lfloor p_r - (w_r - (\hat{c} - w')) p_{r-1} / w_{r-1} \rfloor \right)$
else $u := 0;$
if $z \geq \hat{z} + p' + u$ **then go to 5;**
if $u = 0$ **then go to 4;**

3. [Guardar la solución actual]

$\hat{c} := \hat{c} - w';$
$\hat{z} := \hat{z} + p';$
for $k := j$ **to** r-1 **do** $\hat{x}_k := 1;$
$\overline{w}_j := w';$
$\overline{p}_j := p';$
$\overline{r}_j := r;$
for k := j + 1 **to** r - 1 **do**
  begin
    $\overline{w}_k := \overline{w}_{k-1} - w_{k-1};$ $\overline{p}_k := \overline{p}_{k-1} - p_{k-1};$ $\overline{r}_k := r;$
  end;
for $k := r$ **to** $\tilde{r}$ **do**
  begin
    $\overline{w}_k := 0;$ $\overline{p}_k := 0;$ $\overline{r}_k := k;$
  end;
$\tilde{r} := r$-1;
$j := r + 1;$
if $\hat{c} \geq m_{j-1}$ **then go to 2;**
if $z \geq \hat{z}$ **then go to 5;**
$p' := 0;$

4. [Actualiza la mejor solución]

$z := \hat{z} + p';$
for $k := 1$ **to** j-1 **do** $x_k := \hat{x}_k;$
for $k := j$ **to** r-1 **do** $x_k := 1;$
for $k := r$ **to** n **do** $x_k := 0;$
if $z = U$ **then return;**

continúa.....

.....continuación de procedimiento MT:

5. [Retrocede]

```
hallar  i = max {k < j: x̂_k = 1};
if no existe tal i then return;
ĉ := ĉ + w_i;
ẑ := ẑ - p_i;
x̂_i := 0;
j := i + 1;
if ĉ - w_i ≥ m_i then go to 2;
j := i;
h := i;
```

6. [Tratar de reemplazar el ítem i con el ítem h]

```
h := h + 1;
if z ≥ ẑ + ⌊ĉ p_h / w_h⌋ then go to 5;
if w_h = w_i then go to 6;
if w_h > w_i then
    begin
        if w_h > ĉ or z ≥ ẑ + p_h then go to 6;
        z := ẑ + p_h;
        for k := 1 to n do x_h := x̂_k;
        x_k = 1;
        if z = U then return;
        i := h;
        go to 6;
    end;
else
    begin
        if ĉ - w_h < m_h then go to 6;
        ĉ := ĉ - w_h;
        ẑ := ẑ + p_h;
        x̂_h := 1;
        j := h + 1;
        w̄_h := w_h;
        p̄_h := p_h;
        r̄_h := h + 1;
        for k := h + 1 to r̄ do
            begin
                w̄_k := 0;
                p̄_k := 0;
                r̄_k := k;
            end;
        r̃ := h;
        go to 2;
    end;
end;
```

Cuadro 2.5. Procedimiento del algoritmo Martello-Toth.

d) La cuarta y última modificación consiste en que las cotas superiores asociadas con los nodos del árbol de decisión se calculan mediante una técnica paramétrica basada en el almacenamiento de la información relacionada con la solución actual. Imaginémonos que la solución actual ha sido construida insertando todos los ítems desde el j hasta r.

Entonces, cuando se efectúa un retroceso sobre uno de los ítems, digamos item i ($j \leq i < r$), si no se han insertado los ítems precedentes al ítem j, es posible insertar cuando menos los items i+1,...,r en la nueva solución actual. Con esta finalidad se introducen las notaciones:

$$\tilde{r} = r-1, \quad \bar{r}_i = r-1, \quad \bar{p}_i = \sum_{k=i}^{r} p_k, \quad \bar{w}_i = \sum_{k=i}^{r} w_k \quad \text{para } i = j,...,r.$$

e) El ***Criterio de parada*** consiste en que el procedimiento finaliza en uno de estos dos supuestos:

   i) Cuando se obtiene la cota superior U=U2; ó bien,

   ii) Cuando no sea posible efectuar ningún retroceso.

El procedimiento detallado del algoritmo MT1, con las mismas notaciones usadas en el algoritmo anterior, se presenta en el Cuadro 2.5.

3.2.3.3 Algoritmos de la programación dinámica en PM

La programación dinámica es un método procedente del campo de la Teoría de Control que consiste en dividir un problema en sub-problemas más simples para resolverlos. Su característica principal es reducir un problema a su caso más simple; resolver este caso más simple y conservar las soluciones factibles de éste para utilizarlas en el siguiente caso, un poco más complejo. Luego progresivamente, sobre la base del problema simple anterior construye un problema más complejo y resuelve con la ayuda de la solución anterior; y así sucesivamente, hasta alcanzar la complejidad del problema inicial.

La programación dinámica, que simula las actividades planificadas por períodos de tiempo (ya que se ejecuta por etapas), fue desarrollada por Bellman [12] en los años 50 como una técnica para resolver procesos de decisión secuencial temporal discreta con aplicaciones fundamentalmente en los problemas de optimización.

Para resolver un problema de optimización mediante la programación dinámica, primeramente se tiene que descomponer en sub-problemas encajados denominados *etapas*, luego, por el principio de optimalidad de Bellman, resolviendo en forma secuencial los sub-problemas asociados con cada etapa, se obtendrá la solución.

Para dar una idea intuitiva de cómo la PD encara un problema de decisión, consideremos un sistema de decisión esquematizado en la Figura 3.1. Como se puede observar, tanto la salida Y como la medida de efectividad R dependen tanto de la entrada X como de la decisión D, lo que escribimos:

$$Y = F(X,D), \qquad R = r(X,D),$$

El objetivo de todo sistema de decisión es elegir la mejor decisión posible, lo que se refleja cuando se consigue optimizar la medida de efectividad, es decir,

$$h(X) = opt(r(X,D)).$$

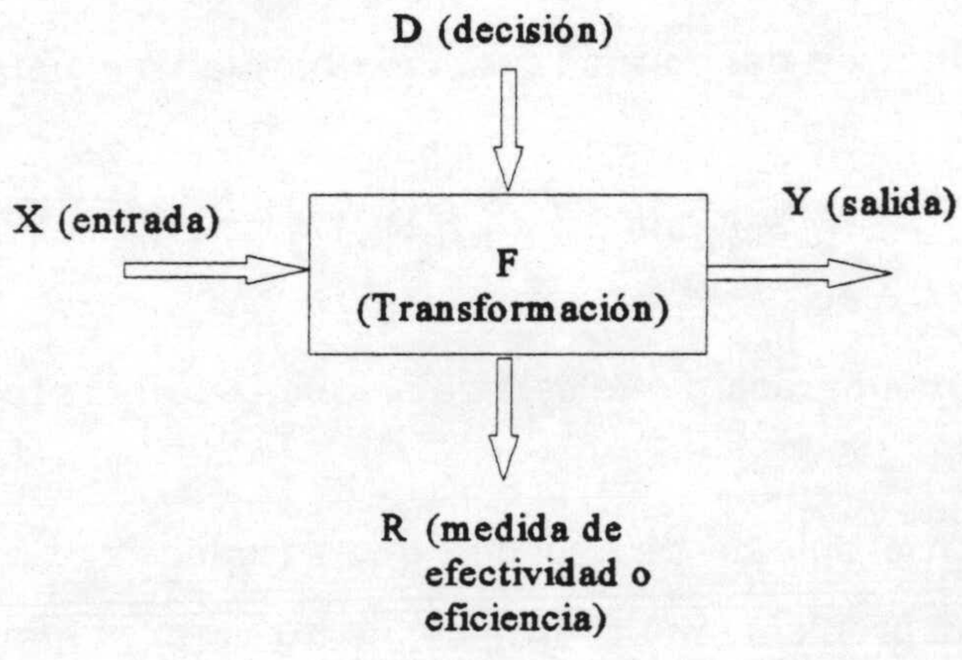


Figura 3.1. Sistema de decisión de una etapa

Consideremos que Y=F(X,D)=G(X), es decir, la dependencia de Y con respecto a D está implícita en la transformación G. La función G se puede descomponer, expresándose entonces como composición de varias funciones g_k (k=1,...,n) , así:

$$Y=G(X)=g_n(g_{n-1}\,(...(g_2\,(g_1(X)))...)).$$

Sea

$$X_k = g_k(X_{k-1})=f_k(X_{k-1},D_k)$$

la salida de la k-ésima etapa con entrada X_{k-1} , con decisión D_k y con transformación f_k, y sea

$$R_k=r_k(X_{k-1},D_k)$$

su correspondiente medida de efectividad.

Entonces el sistema de decisión inicial queda descompuesto en n etapas para k=1,2,...,n respectivamente, lo que da base al principio de optimalidad de Bellman.

En el anterior proceso la descomposición de la transformación F es evidente, pero la descomposición de la medida de efectividad R no es muy clara, esto hace que no todo problema puede ser encarado mediante la PD, sino sólo aquellas que satisfacen con el principio de optimalidad de Bellman. Este principio afirma que *en el sistema descompuesto en n etapas si las decisiones* $D_1,...,D_{n-1}$ *son las óptimas en sus respectivas etapas, la decisión óptima en la n-ésima etapa genera un resultado óptimo de todo el sistema.* Este principio significa que existe una función denominada *función recursiva* o simplemente *recursión*, tal que:

$$h_k(X_k)=opt_{Dk}\ (r_k(X_{k-1},D_k)*h_{k-1}(X_{k-1}))$$

donde * representa alguna operación que está relacionada con el problema en concreto.

De esta manera un problema de decisión que satisface el principio de optimalidad, queda transformado en un problema multi-etápico, de tal manera que en cada etapa se tiene un problema fácil de resolver y resolviendo todas las etapas se tiene la solución del problema inicial.

La aplicación de la programación dinámica al PM nace casi a la par que la propia programación dinámica, en los trabajos iniciales de Bellman, aplicación que luego sería complementada con la aportación de Dantzig.

La descomposición natural del PM consiste en los sub-problemas $PM(m,\hat{c})$ con m $(1\le m\le n)$ ítems y con capacidad $\hat{c}$ $(0\le\hat{c}\le c)$.

Sea $f_m(\hat{c})$ el valor de la solución óptima de $PM(m,\hat{c})$, es decir,

$$f_m(\hat{c})=max\left\{\sum_{j=1}^{m}p_jx_j : \sum_{j=1}^{m}w_jx_j\le\hat{c}; x_j\{0,1\}, j=1,..,m\right\},$$

el caso más simple es:

$$f_1(\hat{c}) = \begin{cases} 0 & \hat{c} = 0,\ldots,w_1 - 1 \\ p_1 & \hat{c} = w_1,\ldots,c. \end{cases}$$

Para resolver el PM, la programación dinámica considera n etapas (igual al número de ítems) y calcula en cada una de ellas, los valores $f_m(\hat{c})$, usando la recursión clásica de Bellman-Dantzig siguiente:

$$f_m(\hat{c}) = \begin{cases} f_{m-1}(\hat{c}) \\ max\left(f_{m-1}(\hat{c}),\ f_{m-1}(\hat{c} - w_m) + p_m\right), \quad \hat{c} = w_m,\ldots, c \end{cases} \tag{3.22}$$

Las soluciones factibles $X_{\hat{c}}$ asociadas a los valores $f_m(\hat{c})$ se denominan *estados*. La solución óptima del problema es el estado correspondiente a $f_n(c)$.

La implementación de este algoritmo es de tiempo de ejecución y espacio de almacenamiento muy elevados porque calcula todos los estados incluidos aquéllos que no influyen en la determinación de los estados óptimos. Felizmente este inconveniente se ha menguado con la propuesta de Horowitz y Sahni, que es la de eliminar estos estados denominados *estados dominados.*

Notamos los estados en cada etapa por $(P_{\hat{c}},\ X_{\hat{c}})$, donde $P_{\hat{c}} = f_m(\hat{c})$ y $X_{\hat{c}}$ es la solución factible. Los *estados dominados* son aquellos estados $(P_{\hat{c}},\ X_{\hat{c}})$ para los cuales existe un estado $(P_y,\ X_y)$ tal que $P_y \geq P_{\hat{c}}$ y $y < \hat{c}$; esto significa que cualquier solución que se obtenga de $(P_{\hat{c}},\ X_{\hat{c}})$ puede ser obtenida de $(P_y,\ X_y)$ por lo que no es necesario calcularla ni reservarla.

Con esta modificación se obtiene el denominado **algoritmo dinámico con estados reducidos (ADER)** que en cada etapa calcula y reserva los estados no dominados, los mismos que utiliza como valores de entrada para determinar los estados de la siguiente etapa.

El procedimiento detallado de este algoritmo aparece en dos procesos: PRODIN (Cuadro 3.6) y RECUR (Cuadro 3.7). El primero inicializa el proceso y luego divide el problema en n etapas; posteriormente, en cada etapa recurre al proceso RECUR para

determinar los estados no dominados. En dichos procedimientos se utilizan las siguientes notaciones:

s: número de estados en la etapa m-1; (3.23)

b: con valor 2^{m-1} (3.24)

$W1_i$: peso total del i-ésimo estado (i=1,...,s) (3.25)

$P1_i$: valor total del i-ésimo estado (i=1,...,s) (3.26)

$X1_i=\{x_{m-1},x_{m-2},...,x_1\}$ (3.27)

donde x_j define el valor de la j-ésima variable en la solución óptima parcial del i-ésimo estado, es decir,

$$W1_i = \sum_{j=1}^{m-1} w_j x_j \quad y \quad P1_i = \sum_{j=1}^{m-1} p_j x_j$$

las componentes de W1 y P1 están estrictamente ordenadas de menor a mayor.

Procedimiento **PRODIN**

```
input: n, c, (p_i), (w_i);
output; z, (x_j);
begin
    W1_0 := 0;
    P1_0 := 0;
    X1_0 := 0;
    s := 1;
    b := 2;
    W1_1 := w_1;
    P1_1 := p_1;
    X1_1 := 1;
    for m: = 2 to n do
        begin
            call RECUR;
            for k = 0 to s do
                begin
                    W1_k := W2_k;
                    P1_k := P2_k;
                    X1_k := X2_k;
                end;
        end;
    z := P1_s;
    determine (x_j) mediante la decodificación de X1_1.
end;
```

Cuadro 3.6. Procedimiento del algoritmo de la PD.

Procedimiento RECUR

```
input:s ,b,(W1_i), (P1_i) (X1_i),w_m, p_m, c;
output; s,b,(W2_k), (P2_k), (X2_k);
begin
        i := 0;
        k := 0;
        h := 0;
        y := w_m ;
        W1_{s+1} := +∞ ;
        W2_0 := 0;
        P2_0 := 0;
        X2_0 := 0;
while  min (y,W1_h) ≤ c do
      if W1_h ≤ y  then
            begin
                  Comentario   : define el  siguiente  estado (p,  x)
                  p := P1_h ;
                  x := X1_h ;
                  if W1_h ≤ y then
                        begin
                              if  P1_h + p_m  > p then
                                    begin
                                          p := P1_i + p_m ;
                                          x := X1_i + b;
                                    end;
                              i := i + 1;
                              y := W1_i + w_m ;
                        end;
                  Comentario   : guarda  el siguiente  estado  no dominado .
                  if  p > P2_k  then
                        begin
                              k := k + 1;
                              W2_k  := W1_h ;
                              P2_k  := p;
                              X2_k  := x;
                        end;
                  h := h + 1;
            end;
      else
            begin
                  Comentario   : guarda  el nuevo  estado  no dominado .
                  if  P1_i + p_m  > P2_k  then
                        begin
                              k := k + 1;
                              W2_k  := y;
                              P2_k  := P1_i + p_m ;
                              X2_k  := X1_i + b;
                        end;
                  i: = i + 1;
                  y: = W1_1 + w_m ;
            end;
      s := k;
      b := 2b;
end;
```

Cuadro 3.7. Parte recursiva, que calcula los estados no dominados.

En los procedimientos se utilizan los índices *i* para revisar los estados de la etapa actual y el índice k para guardar los estados de la nueva etapa. Cada estado actual puede producir un nuevo estado de peso total $y=W1_i+w_{m;}$ así los estados actuales son de peso total $W1_h<y$, y el nuevo estado se guarda en la nueva etapa siempre que sea no dominado por algún estado ya guardado. Después de la ejecución de RECUR los valores (3.23) y (3.24) son relativos a la nueva etapa, mientras que los nuevos valores de (3.25), (3.26) y (3.27) son dados por $(W2_k)$, $(P2_k)$ y $(X2_k)$, respectivamente. Además, $X1_i$ y $X2_k$, utilizados en los procesos PRODIN y RECUR son números enteros cuya representación binaria es la concatenación de los elementos del conjunto dado en (3.27), es decir, son números cuya representación binaria es la concatenación de $x_{m-1},x_{m-2,...,}x_1$.

Por otro lado, además de proponer la eliminación de los estados no dominados en la solución del PM mediante la programación dinámica, Horowitz y Sahni propusieron un algoritmo basado en la subdivisión del problema original de *n* variables en dos sub-problemas independientes, cada uno con q y r=n-q variables, donde $q=\lceil n/2 \rceil$. Este algoritmo se denomina **algoritmo dinámico de Horowitz-Sahni (HS2)**.

HS2, para resolver ambos sub-problemas, utiliza los procesos PRODIN y RECUR de los Cuadros 3.6 y 3.7 respectivamente, obteniendose así una lista de estados para cada sub-problema. Para obtener la solución del problema global combina las listas correspondientes a la última etapa de ambos sub-problemas.

3.2.3.4 Algoritmo de reducción de Martello y Toth

Las variables de toda solución factible de un PM se pueden particionar en dos conjuntos, uno formado por variables con valor 1, y otro con variable cuyo valor es cero. Con ello se ha particionado el conjunto $N = \{ 1,2,...,n \}$ en dos conjuntos: $N1 = \{ j \in N: x_j = 1 \}$ y $N0 = \{ j \in N: x_j = 0 \}$.

Consideremos ahora el conjunto de las soluciones exactas obtenidas mediante diferentes algoritmos. Hay que tener en cuenta que dichas soluciones no son necesariamente óptimas ni mucho menos todas iguales. Sin embargo, posiblemente existen variables que en todas las soluciones exactas tienen el mismo valor, por tanto podemos realizar la siguiente

partición de N:

$$J1 = \{ j \in N: x_j = 1 \text{ en todas las soluciones exactas de PM} \}$$

$$J0 = \{ j \in N: x_j = 0 \text{ en todas las soluciones exactas de PM} \}$$

$$F = N \setminus (J1 \cup J0).$$

Si se conoce tanto J1 y J0, el PM se transforma en el problema más simple siguiente:

$$\max \quad z = \sum_{j \in F} p_j x_j + \hat{p}$$

$$s.a. \quad \sum_{j \in F} w_j x_j \leq \hat{c}$$

$$x_j \in \{0,1\} \quad j \in F,$$

donde $\hat{p} = \sum_{j \in J1} p_j$, $\hat{c} = c - \sum_{j \in J1} w_j$.

Ingargiola y Korsh fueron los primeros en proponer un método para determinar J1 y J0. La idea básica de dicho método es la siguiente:

Si x_j = b (b=0, ó 1) produce infactibilidad o implica una peor solución que la mejor solución existente, entonces x_j tendrá el valor 1-b en cualquier solución exacta. Sea l el valor de una solución factible de PM, y para $j \in N$, sea:

$$u_j^l \quad \text{(respectivamente } u_j^0\text{)} \tag{3.28}$$

un límite superior para PM con la restricción adicional $x_j = 1$(respectivamente $x_j = 0$). Entonces tenemos:

$$J1 = \left\{ j \in N : u_j^0 < l \right\},$$
$$J0 = \left\{ j \in N : u_j^1 < l \right\}. \tag{3.29}$$

En el algoritmo de Ingargiola-Korsh, las cotas superiores (3.28) son calculadas mediante la cota superior de Dantzig.

Existen otros algoritmos de reducción que mejoran sustancialmente el algoritmo de Ingargiola-Korsh, de entre los que se destaca por su eficiente el algoritmo propuesto por Martello y Toth [132,133], y las diferencias sustanciales en los procedimientos serían las siguientes:

a) Las cotas superiores de (3.28) son calculadas mediante la cota U_2 de Martello-Toth.

b) J1 y J0 son determinados al final, usando la mejor solución heurística l.

c) El valor crítico s es calculado mediante la búsqueda binaria.

El procedimiento detallado de este algoritmo de reducción se presenta en el Cuadro 3.8, mediante el procedimiento denominado REDUC.

Procedimiento **REDUC**

input: n, c, (p_i), (w_i);
output: J1, J0, l;
begin
  for $j := 0$ **to** n **do**
    begin
      $\overline{p}_j = \sum_{i=1}^{j} p_i$;
      $\overline{w}_j = \sum_{i=1}^{j} w_i$;
    end;
  $s = min\{j : \overline{w}_{j-1} \le c < \overline{w}_j\}$;
  $l := \overline{p}_{s-1}$;
  $\overline{c} := c - \overline{w}_{s-1}$;
  for $j := s+1$ **to** n **do**
    if $w_j \le \overline{c}$ **then**
      begin
        $l := l + p_j$;
        $\overline{c} := \overline{c} - w_j$;
      end;
for $j := 1$ **to** s **do**
  begin
    hallar $\overline{s}$ *tal que* $\overline{w}_{\overline{s}-1} \le c + w_j < \overline{w}_{\overline{s}}$;
    $\overline{c} := c + w_j - \overline{w}_{\overline{s}-1}$;
    $u_j^0 := \overline{p}_{\overline{s}-1} - p_j + max(\lfloor \overline{c} p_{\overline{s}+1} / w_{\overline{s}+1} \rfloor, \lfloor p_{\overline{s}} - (w_{\overline{s}} - \overline{c}) p_{\overline{s}-1} / w_{\overline{s}-1} \rfloor)$;
    $l := max(l, \overline{p}_{\overline{s}-1} + p_j)$;
  end;
for $j := s$ **to** n **do**
  begin
    hallar $\overline{s}$ *tal que* $\overline{w}_{\overline{s}-1} \le c - w_j < \overline{w}_{\overline{s}}$;
    $\overline{c} := c - w_j - \overline{w}_{\overline{s}-1}$;
    $u_j^1 := \overline{p}_{\overline{s}-1} + p_j + max(\lfloor \overline{c} p_{\overline{s}+1} / w_{\overline{s}+1} \rfloor, \lfloor p_{\overline{s}} - (w_{\overline{s}} - \overline{c}) p_{\overline{s}-1} / w_{\overline{s}-1} \rfloor)$;
    $l := max(l, \overline{p}_{\overline{s}-1} - p_j)$;
  $J1 := \{j \le s : u_j^0 \le l\}$;
  $J0 := \{j \ge s : u_j^1 > l\}$;
end;

Cuadro 3.8. El procedimiento del algoritmo de reducción de Martello y Toth.

3.2.4 Criterios de parada difusos en los algoritmos del PM

La complejidad de los algoritmos más eficientes para resolver el PM es como promedio $O(n^2)$ en el caso de los algoritmos de ramificación, y O(nc) en el caso de los algoritmos de la programación dinámica que hemos revisado. Si bien es verdad que esta complejidad no es exagerada, dado que se trata de un problema NP-completo, en casos de ejemplares con muchos ítems, el tiempo de ejecución de uno de estos algoritmos para obtener una solución exacta puede ser muy prolongado.

Puesto que en muchas situaciones se admite como aceptable una solución no óptima con tal de que se obtenga en un tiempo razonable, los algoritmos exactos del problema de la mochila pueden ser flexibilizados de tal manera que se puede detener el proceso de cálculo cuando se haya obtenido una solución aceptable.

Como hemos expuesto en el capítulo 2, la flexibilización de los algoritmos exactos se consigue introduciendo un criterio de parada difuso, y esto a su vez, mediante una función de pertenencia.

Los criterios de parada de los algoritmos del PM difieren con respecto a los criterios de parada de los algoritmos de la programación lineal. Al menos esto es lo que ocurre en los algoritmos que hemos revisado que, al igual que otros algoritmos que resuelven el PM, presentan un criterio de parada implícita y no en forma explícita. Esta diferencia nos obliga a definir de otra forma los criterios de parada difusos.

Al flexibilizar un algoritmo de tal manera que se obtenga una solución aceptable, en cierto modo se da lugar a que el número de iteraciones, y por ende el tiempo de ejecución, sea dependiente de la calidad del valor de la solución factible que se obtenga. Dicho de otro modo, el procedimiento del algoritmo flexibilizado se detendrá cuando encuentre una solución factible cuyo valor sea considerado aceptable por el decisor. La calidad de *aceptable* de una solución no óptima la decidirá únicamente el decisor, pero para ello éste deberá tener alguna referencia que le facilite la elección.

Dicha referencia que proponemos, estaría constituida por una cota inferior L_0 y una cota superior U_0 para el valor z^* de la solución óptima del problema de la mochila, es decir,

$$L_0 \leq z^* \leq U_0.$$

Estas cotas tienen que ser obtenidas para cada ejemplar que se pretenda resolver. Como cota superior, por ejemplo, puede utilizarse cualquiera de las cotas superiores estudiadas en la sección 3.2.2, y como cota inferior, por ejemplo, la solución obtenida mediante el algoritmo voraz ó la suma de todos los valores menores que el ítem crítico. Es recomendable utilizar la cota de Dantzig U_1 para la cota superior mientras que para la cota inferior se puede utilizar cualquiera de las formas sugeridas. Así, el tiempo de cálculo de L_0 y U_0 sería a lo sumo de $O(n)$. Usar otras formas de cálculo de la cota superior sólo incrementaría el tiempo de ejecución, que es precisamente lo que se quiere evitar con la introducción del criterio de parada difuso.

Cuando se conocen solamente la cota inferior y superior del valor de la solución óptima, este valor puede expresarse como un número difuso, mediante una función de pertenencia, de la siguiente forma:

$$\mu(z) = \begin{cases} 0 & z < L_0, \\ f(z) & L_0 \leq z \leq U_0. \end{cases} \tag{3.30}$$

donde $f(.)$ es una función continua no decreciente con valor entre [0,1].

Sobre la base de las cotas y la función de pertenencia, si el decisor considerara aceptable la solución cuyo valor tiene un grado de pertenencia mayor que α,, entonces el criterio de parada sería:

$$f(z) \geq \alpha \quad \text{ó} \quad z \geq f^1(\alpha) \tag{3.31}$$

Al introducir este criterio de parada en los algoritmos estudiados, se flexibilizarán de tal manera que se pueda detener el proceso de cálculo cuando se obtenga la condición (3.31).

Una vez fijado el grado de aceptación α existen dos posibilidades:

a) Que el valor z* sea mayor que $f^1(\alpha)$, en cuyo caso se obtendrá una solución aceptable.

b) Que el valor óptimo z* sea menor o igual que $f^1(\alpha)$, en cuyo caso el criterio de parada difusa no tendrá sentido ya que el proceso nunca terminaría, razón por la cual es

necesario mantener los criterios exactos, y que el criterio difuso sea adicional.

Una forma de reducir la posibilidad de que se presente el caso (b) es utilizando una función $f(.)$ cóncava. Como vemos en la Figura 3.2, se cumple $f^{-1}(\alpha) < h^{-1}(\alpha)$ cuando f es cóncava pero h no lo es. La importancia de esta característica radica en que existe grandes posibilidades de evitar el caso (b) cuando se utiliza una función cóncava f en la definición de la pertenencia (3.30), mientras que si se utiliza una función no cóncava como h de la Figura 3.2 hay pocas posibilidades de evitar que se cumpla el caso (b).

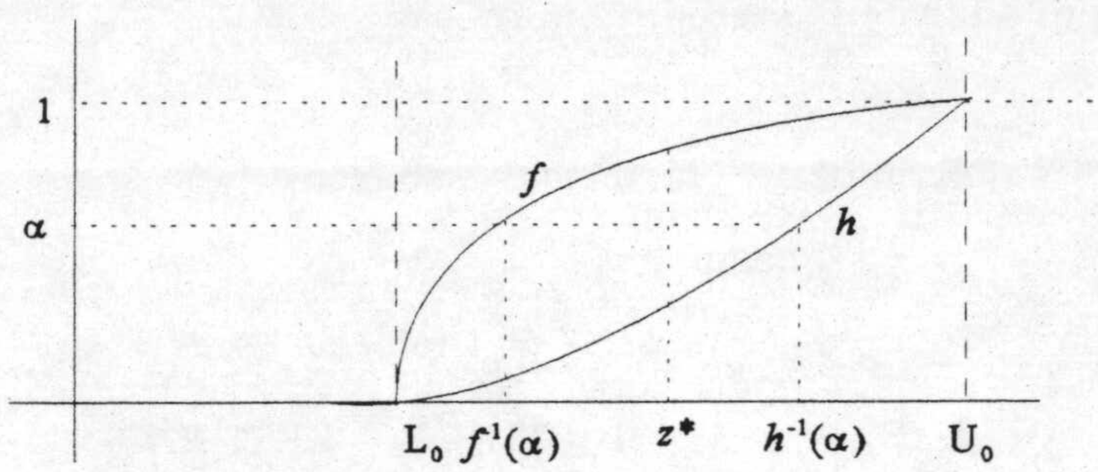


Figura 3.2. Funciones de pertenencia cóncava y no cóncava.

Una función cóncava que se puede utilizar en la definición de la función de pertenencia (3.30) puede ser de la siguiente forma:

$$f(t) = \sqrt[n]{\frac{t - L_0}{U_0 - L_0}} \tag{3.32}$$

donde n>1. Esta función tienen inversa, ya que es inyectiva en el dominio de definición de la función de pertenencia (3.30).

Luego usando (3.32), la condición (3.31) se transforma en:

$$z \geq U_0 + (U_0 - L_0)\alpha^n \tag{3.33}$$

A continuación modificamos los criterios de parada de los algoritmos que hemos revisado en las secciones (3.2.3.2) y (3.2.3.3).

3.2.4.1 Criterios de parada difusos en los algoritmos de RA en el PM

En el algoritmo HS1, el criterio de parada consiste en detener el procedimiento cuando no existe ninguna posibilidad de retroceder. Al flexibilizar el algoritmo, además de mantener el criterio anterior, incluimos la condición (3.31).

Con este criterio de parada adicional, en el procedimiento correspondiente que se encuentra en el Cuadro 3.4, inmediatamente después de actualizar la mejor solución se debe analizar si cumple o no con la condición (3.31), es decir, el número 4 del procedimiento HS del Cuadro 3.4 debe quedar de la siguiente manera:

```
4. [Actualiza    la mejor    solución]
    if  ẑ > z then
        begin
            z := ẑ;
            for  k := 1 to n do   x_k := x̂_k ;
        end;
    IF   z ≥ f^-1(α) THEN    RETURN
    ⋮
```

La instrucción que aparece con letras mayúsculas corresponde al criterio de parada difuso.

En el caso del algoritmo MT1 los criterios de parada son explícitos. El procedimiento se detiene cuando alcanza la cota U_2 o cuando no existe ningún retroceso posible.

En la flexibilización de este algoritmo modificamos la condición: *si* $z=U_2$ *, detener el proceso*, por la condición *si se cumple (3.31), detener el proceso*, es decir, en el procedimiento MT del Cuadro 3.5, tanto en el número 4 como en el número 6, la instrucción:

```
if z = U then return;
```

debemos sustituirla por la instrucción

```
if z ≥ f^1(α) then return
```

Así obtenemos el algoritmo de MT1 con criterio de parada difuso.

Al igual que en el caso anterior, también es este algoritmo se mantiene la condición de detener el proceso cuando no existe ningún retroceso posible.

3.2.4.2 Criterios de parada difusos en los algoritmos de PD en el PM

En primer lugar veamos el criterio de parada en el ADER. En este algoritmo no es explícito el criterio de parada, pero es evidente que el procedimiento se detiene cuando se obtiene el estado con el valor $f_n(c)$. Con este criterio, el procedimiento seguirá hasta el último estado de la última etapa, aunque la solución exacta se haya obtenido en las etapas anteriores o estados anteriores.

Al introducir el criterio de parada difusa, en cada etapa y en cada estado se analizará si se ha obtenido un valor aceptable o no; si es afirmativo, se detiene el proceso. Naturalmente, con este criterio no hay la necesidad de resolver todas las etapas ni todos los estados.

En consecuencia, el procedimiento de este algoritmo dado en los Cuadros 3.6 y 3.7 quedaría modificado de la siguiente manera:

- En el procedimiento PRODIN:

```
                    ⋮
                    begin
                        W1_k := W2_k;
                        P1_k := P2_k;
                        X1_k := X2_k;
                    end;
                IF P1_s ≥ f^-1(α) THEN BREAK;
            end;
            ⋮
```

- En el procedimiento RECUR:

```
                        ⋮
                    s := k;
                    b := 2b;
                    IF P2_S ≥ f^-1(α) THEN BREAK;
                end;
```

Con ambos procedimientos se obtiene el algoritmo dinámico con estados reducidos y con criterio de parada difuso.

En el caso del algoritmo HS2, que divide el problema en dos sub-problemas y los resuelve cada uno usando ADER para finalmente combinar los resultados obtenidos, la flexibilización, adicionalmente a la heredada de la flexibilización de ADER, tiene que hacerse en la parte de la combinación de las listas de los resultados de los dos subproblemas

imponiendo la condición (3.31). Con este último criterio adicional, la combinación de las listas finales no se hará con todos los resultados sino solo hasta que cumpla con la condición (3.31).

3.2.5 Experimentos computacionales

En esta sección presentamos los resultados de los experimentos computacionales realizados con problemas de la mochila de 1.000, 5.000, 10.000 y 50.000 ítems cuyos valores y pesos comprendidos en el intervalo [1,1000] han sido generados aleatoriamente y como capacidad de la mochila se ha utilizado la semi-suma de los pesos.

En cada uno de los algoritmos con criterios de parada difusos vistos en la sección anterior se considera un grado de pertenencia α=0.5, α=0.8 y α=1; este último caso corresponde a la solución obtenida mediante el algoritmo sin el criterio de parada difuso.

En cada una de las Tablas de los resultados que corresponden a cada uno de los algoritmos con criterios de parada difusos vistos en la sección 3.2.4, para cada valor α indicado, se consignan el error cometido y el tiempo de ejecución. El error se obtiene mediante la siguiente fórmula:

$$error = \frac{(Cota\ de\ Dantzig) - (solución\ obtenida)}{cota\ de\ Dantzig} * 100$$

Debido a que los algoritmos con criterios de parada difusos son algoritmos de aproximación, para comparar su ventaja con respecto al algoritmo aproximado de Sahni se muestran también los resultados que proporciona dicho algoritmo tanto para k=2 como para k=3.

En los criterios de parada difusos de los cuatro algoritmos revisados en la sección 3.2.4, cuyos resultados presentamos en esta sección, se ha utilizado la condición:

$$z \geq L_0 + (U_0 - L_0).\alpha^4$$

es decir, los criterios difusos están definidos mediante la función de pertenencia (3.30) con f(.) dada por (3.32) con n=4 (un valor no muy grande pero que le da buena concavidad a la función f); L_0 es la solución respectiva obtenida mediante el algoritmo voraz y U_0 es la cota de Dantzig.

Las Tablas 3.1, 3.2, 3.3 y 3.4 muestran los resultados de los experimentos computacionales tras haber utilizado los algoritmos con criterios de parada difusos de la sección 3.2.4, y en cada una de ellas, para poder compararlas, se consignan los resultados que proporciona el algoritmo aproximado de Sahni .

n	Algoritmo HS1						Algoritmo de Sahni			
	exacta		α=0.5		α=0.8		k=2		k=3	
	error	tiempo	error	tiempo	error	tiempo	error	tiempo	error	tiempo
1.000	.001928	.044	.00517	.0	.003959	.0	.0032556	.11	.002471	.23
5.000	.000098	1.462	.000464	.0	.000316	.022	.000237	2.538	.000168	4.58
10.000	.00002	3.72	.000108	.01	.000059	.012	.000298	8.802	.000272	13.97
50.000	.00000	21.8380	.000021	.096	.000013	.136	.000003	212.476	.0	257.764

Tabla 3.1. Algoritmo de Horowitz-Sahni con criterio de parada difuso.

n	Algoritmo Martello-Toth (RA)						Algoritmo de Sahni (aprox.)			
	Exacta		α=0.5		α=0.8		k=2		k=3	
	error	tiempo	error	tiempo	error	tiempo	error	tiempo	error	tiempo
1.000	.001928	.042	.00517	.03	.003959	.04	.0032556	.11	.002471	.23
5.000	.000098	1.462	.000464	1.407	.000316	1.446	.000237	2.538	.000168	4.58
10.000	.00002	4.736	.000098	4.624	.000059	4.712	.000298	8.802	.000272	13.97
50.000	.0	114.324	.000022	114.114	.000013	114.312	.000003	212.476	.0	257.764

Tabla 3.2. Algoritmo de Martello-Toth con criterio de parada difuso.

Como podemos observar en la Tabla 3.1, el algoritmo de Horowitz Sahni (método de ramificación y acotación) con criterio de parada difuso proporciona resultados con muy pocos errores y en tiempos también muy cortos tanto con respecto a la solución exacta como con respecto a soluciones aproximadas dadas por el algoritmo de Sahni. La ventaja en el tiempo de ejecución de la versión difusa del algoritmo con respecto a los tiempos de ejecución del

caso no difuso y con respecto al algoritmo aproximado de Sahni es muy grande, haciéndose más significativa cuando n (número de ítems) se incrementa.

Asimismo, el algoritmo de Martello-Toth con criterio de parada difuso (del método de ramificación y acotación), es más ventajoso en cuanto al tiempo de ejecución respecto al algoritmo aproximado de Sahni, esta ventaja se hace más significativa cuando n se incrementa; aunque supone sólo una leve mejoría con respecto a la solución exacta, como podemos observar en la Tabla 3.2.

Las ventajas que se observan en el tiempo de ejecución de los algoritmos con criterios de parada difusos alcanzan mayor relevancia si tenemos en cuenta que el algoritmo aproximado de Sahni se ha aplicado *después* de reducir (mediante el procedimiento REDUC Cuadro 3.8.) el problema inicial, reducción que consigue una disminución considerable de tiempo de ejecución.

Con relación a los resultados de las Tablas 3.3.y 3.4, la escasa diferencia que existe en los tiempos de ejecución se debe a que se ha utilizado primero el algoritmo de reducción de Martello-Toth (procedimiento REDUC Cuadro 3.8), y luego, se ha aplicado solamente en el problema reducido, tanto los algoritmos con criterios de parada difusos como el algoritmo aproximado de Sahni. Aún así, se observa la ventaja en el tiempo de ejecución de los algoritmos con criterios de parada difusos con respecto los algoritmos correspondiente sin los criterios difusos, y con respecto al algoritmo aproximado de Sahni, se aprecia la ventaja en el error correspondiente.

n	Algoritmo dinámico con estados reducidos						Algoritmo de Sahni (aprox.)			
	Exacta		α=0.5		α=0.8		k=2		k=3	
	error	tiempo	error	tiempo	error	tiempo	error	tiempo	error	tiempo
1.000	.001928	.24	.004367	.19	.00382	.236	.0032556	.11	.002471	.23
5.000	.000098	3.778	.000276	3.076	.000247	3.098	.000237	2.538	.000168	4.58
10.000	.00002	12.198	.000069	10.182	.000029	10.252	.000298	8.802	.000272	13.97
50.000	.0	223.944	.000004	218.350	.000003	218.376	.000003	212.476	.0	257.764

Tabla 3.3. Algoritmo dinámico con estados reducidos con criterio de parada difuso.

n	Algoritmo de Horowitz-Sahni (PRODIN)						Algoritmo de Sahni (aprox.)			
	Exacta		α=0.5		α=0.8		k=2		k=3	
	error	tiempo	error	tiempo	error	tiempo	error	tiempo	error	tiempo
1.000	.002576	.67	.006421	.428	.004398	.484	.0032556	.11	.002471	.23
5.000	.000177	12.978	.000672	11.534	.000395	12.918	.000237	2.538	.000168	4.58
10.000	.000024	61.364	.000118	41.688	.000029	41.976	.000298	8.802	.000272	13.97
50.000	.0	488.136	.000016	483.292	.000011	485.070	.000003	212.476	.0	257.764

Tabla 3.4. Algoritmo de Horowitz-Sahni (PD) con criterio de parada difuso.

En todos los casos en las que se ha utilizado el algoritmo de reducción de Martello-Toth, el problema reducido que se obtiene tiene sólo 10 ítems, sobre este nuevo problema se ha aplicado los correspondientes algoritmos. En consecuencia, la diferencia en el tiempo así como en el error obtenido en las Tablas 3.3 y 3.4 se reflejan sólo la diferencia en el nuevo problema.

3.3 EL PROBLEMA DEL VIAJANTE DE COMERCIO

El TSP, pese a que según Hoffman y Wolfe [83] los orígenes de este problema se encuentran en los planteamientos tanto de Euler como de Vandermonde (siglo XVIII) sobre la ruta del caballo en un tablero de ajedrez, en el mundo de las matemáticas recién a principios de la década 50 alcanzó gran popularidad debido a su relación con prominentes problemas combinatoriales que dieron origen a nuevas ramas de la programación lineal como son los problemas de asignación y problemas de transporte.

Actualmente, no existe ninguna discusión sobre la importancia del TSP, por una parte por sus múltiples aplicaciones como se puede ver en [20, 44, 60, 68, 112, 118] entre otros; y por otra parte porque el TSP, debido a su característica de NP-completo entre otras razones, constituye un banco de pruebas de diversos algoritmos ya sean exactos, aproximados o heurísticas, razón por la cual no podíamos dejar de verificar nuestra

propuesta en este problema.

Nuestra propuesta de nuevo consiste en introducir criterios de parada difusos en los algoritmos exactos para el TSP, para que puedan ser utilizados en la obtención de soluciones aceptables para el decisor.

En el TSP los algoritmos exactos generales más eficientes se derivan de las aplicaciones del método de ramificación y acotación, razón por la cual en tales algoritmos introduciremos los criterios de parada difusos en la sección 3.3.4. Previamente en la sección 3.3.1 presentamos la formulación matemática, en la sección 3.3.2 el método de ramificación y acotación en el TSP y en la sección 3.3.3 cotas para el valor óptimo.

3.3.1 Formulación y tipos de TSP

Comenzamos presentando la formulación matemática del TSP con n ciudades, incluida la ciudad desde donde se inicia el recorrido. Dicha ciudad se acostumbra a notar i=1 y las demás n-1 ciudades como ciudad 2, 3,...n, además utilizamos las notaciones siguientes:

d_{ij} = la distancia desde la ciudad i a la ciudad j ($i \neq j$)

$$x_{ij} = \begin{cases} 1 & \text{si en el trayecto del viajante a la ciudad } i \text{ le sigue la ciudad } j. \\ 0 & \text{en otros casos} \end{cases}$$

Puesto que la aspiración del viajante es recorrer la menor distancia posible, la función objetivo será:

$$\min \quad z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \tag{3.34}$$

Para garantizar que cada ciudad j sea visitada una única vez, debemos imponer dos condiciones, la primera

$$\sum_{i=1}^{n} x_{ij} = 1; \quad j = 1,2,\ldots,n \tag{3.35}$$

para indicar que a cada ciudad el viajante llega una sola vez y la segunda condición

$$\sum_{j=1}^{n} x_{ij} = 1; \quad i = 1,2,\ldots,n \tag{3.36}$$

para indicar que el viajante sale una sola vez de cada ciudad.

Si utilizamos el lenguaje de grafos para representar el TSP, los nodos como ciudades y los caminos como arcos, la condición (3.35) indica que para cada ciudad hay un sólo arco de llegada, mientras que la condición (3.36) indica que hay un sólo arco de salida.

Adicionalmente, debido a que las dos condiciones anteriores no evitan que se generen sub-rutas aisladas unas de otras, se deben incluir otras restricciones. Un conjunto de ciudades $Q \subset N$ ($N = \{1,2,...,n\}$) forman una sub-ruta si en el recorrido el viajante no puede salir de dicho grupo de ciudades, esto se puede evitar introduciendo un arco que una la ciudad i de Q con cualquier ciudad de N-Q; en general se evitará una sub-ruta si imponemos la condición de que siempre alguna ciudad i de todo subconjunto Q estará unido con alguna ciudad j de N-Q, esto es:

$$\sum_{i \in Q} \sum_{j \in \bar{Q}} x_{ij} \geq 1, \qquad \forall\ Q \subset N,\ \ \bar{Q} = N - Q \tag{3.37}$$

o equivalentemente

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1, \qquad \forall\ Q \subset N \tag{3.38}$$

esta última restricción indica que en cada subconjunto $Q \subset N$ el número total de arcos debe ser menor que el número de ciudades $|Q|$.

Finalmente el modelo como un problema de PL de TSP es:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij} \\ \text{s. a.} \quad & \sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,..., n, \\ & \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,..., n, \\ & \sum_{i \in Q} \sum_{j \in \bar{Q}} x_{ij} \geq 1 \qquad \forall\, Q \subset N \\ & x_{ij} \in \{0,1\} \qquad i, j = 1,2,..., n; \end{aligned} \tag{3.39}$$

donde, cuando $i=j$ se asume que d_{ii} es suficientemente grande para obligar que en el óptimo $x_{ii}=0$.

Si las n ciudades del TSP se encuentran localizadas en un mismo plano geométrico, al correspondiente problema se le denomina TSP *Euclidiano,* y las distancias entre ellas están dadas por la distancia Euclidiana. Generalmente un problema de TSP se le identifica mediante una matriz cuadrada de orden n, $d = (d_{ij})$, donde d_{ij} representa la distancia de la ciudad i a la ciudad j y n el número de ciudades. De acuerdo a las características de su matriz de las distancias existen diferentes tipos de casos especiales de TSP. En tales casos, como por ejemplo cuando d es: una matriz triangular superior ($d_{ij} = 0$ si $i > j$), una matriz graduada ($d_{ij} \leq d_{i,j+1}$ y/o $d_{ij} \geq d_{ij}$ para todo i,j), etc., existen métodos muy eficientes. Una buena recopilación de estos y otros TSP especiales podemos encontrarla en [27, 72]. Otros dos grupos importantes de TSP, que no son excluyentes con los casos anteriores, son aquellos cuyas matrices de distancias son simétricas y asimétricas, denominados respectivamente TSP simétrico y TSP asimétrico, un caso particular de TSP con matrices simétricas es el TSP Euclidiano.

Además de su representación como un problema de programación lineal, el TSP se puede considerar como un grafo. C on mayor frecuencia, esta representación se utiliza para un TSP simétrico, es decir cuando $d_{ij} = d_{ji}$. Se puede considerar como un grafo completo no dirigido G=(V,E), con V el conjunto de n vértices, E el conjunto de aristas y distancias d_{ij}, planteándose el TSP como un problema de minimización

$$\sum_{i \in V} \sum_{j > i} c_{ij} x_{ij} \tag{3.40}$$

sujeto a: $$\sum_{j<i} x_{ji} + \sum_{j>i} x_{ij} = 2, \qquad i \in V \tag{3.41}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \qquad \forall S \subset V, \quad S \neq \Phi \tag{3.42}$$

$$x_{ij} \in \{0,1\}, \quad i, j \in V, \quad j > i, \tag{3.43}$$

la condición de la eliminación de sub-rutas (3.42) puede ser representado también mediante la siguiente restricción:

$$\sum_{\substack{i \in S \\ j>i}} \sum_{j \in V-S} x_{ij} + \sum_{\substack{i \in V-S \\ j>i}} \sum_{j \in S} x_{ij} \geq 2, \quad \forall \quad S \subset V, \quad S \neq \Phi \tag{3.44}$$

El uso de una u otra forma de representación depende del método y tipo de algoritmo que

se utilice para resolver el correspondiente problema.

3.3.2 El método de ramificación y acotación en TSP

Para encontrar las primeras aplicaciones del método de ramificación y acotación (RA) en la solución del TSP tenemos que remontarnos al año 1954, año en que Dantzig, Fulkerson y Johnson [48] propusieron un método para resolver el TSP. Dicho método tenía las características de lo que ahora se conoce como el *método de ramificación y acotación*, aunque entonces aún no se conocía como un método general para resolver problemas de optimización de características específicas. Desde entonces hasta la actualidad se han desarrollado diferentes tipos de algoritmos de RA para resolver el TSP. Una recopilación de los más importantes podemos encontrarla en [6].

En cualquier caso los procedimientos particulares de los diferentes algoritmos de RA para el TSP, según Balas y Toth [6] pueden ser vistos como variantes de una de las siguientes versiones:

Versión 1:

Paso 1: (*Inicialización*) Poner el TSP en una lista (de subproblemas activos). Inicialice la cota superior U=∞.

Paso 2: (*Selección de un subproblema*): Si la lista es vacía, detener el proceso, la ruta asociada con U es óptima (ó si U=∞, el TSP no tiene solución). En otro caso seleccione un subproblema TSP_i de acuerdo a una regla de selección del subproblema y borrar TSP_i de la lista.

Paso 3: (*Acotación inferior*) Resolver la relajación R_i del TSP_i o hallar la cota $v(R_i)$ de R_i. Sea L_i el valor obtenido:

Si $L_i \geq U$, regresar al paso 2.

Si $L_i < U$, y la solución define una ruta para TSP, reemplazar la mayor ruta existente por esta nueva ruta, y hacer $U=L_i$. Luego ir al paso 5.

Si $L_i < U$ pero la solución no define una ruta de TSP, ir al paso 4.

Paso 4: (*Cota superior: opcional*) Utilice una heurística para encontrar una ruta para TSP.

Si se obtiene una ruta mejor que la existente, reemplazarla ésta y actualizar U.

Paso 5: (*Reducción: opcional*) Eliminar del grafo de TSP_i todos los arcos cuya inclusión en una ruta podría aumentar su valor por sobre el valor de U.

Paso 6: (*Ramificación*) Aplicar una regla de ramificación a TSP_i, es decir, generar nuevos sub-problemas TSP_{i1}....TSP_{iq} y colocarlos en la lista, luego ir al paso 2.

Versión 2:

Paso 1: (*Inicialización*) Igual que en la versión 1, pero aquí hay que resolver R antes de poner el TSP en la lista.

Paso 2. (*Selección de un sub-problema*): Igual que en la versión 1.

Paso 3: (*Acotación superior: opcional*) Igual que en el paso 4 de la versión 1.

Paso 4: (*Reducción: opcional*) Igual que en el paso 5 de la versión 1.

Paso 5: (*Ramificación*) Usar una regla de ramificación para definir el conjunto de sub-problemas $TSP_{i1},...,TSP_{iq}$, generado del sub-problema actual TSP_i,.

Paso 6: (*Acotación inferior*) Si todos los subproblemas a ser generados de TSP_i, de acuerdo a una regla de ramificación, ya existen, ir al paso 2. En otro caso generar el siguiente subproblema TSP_{ij}, definido mediante una regla de ramificación, resuelve la relajación R_{ij} deTSP_{ij}, o hallar $v(R_{ij})$ de R_{ij}. Sea L_{ij} el valor obtenido:

Si $L_{ij} \geq U$, repetir el paso 6.

Si $L_{ij} < U$, y la solución define una ruta para TSP, reemplazar la mejor ruta previa por esta nueva ruta, y hacer $U=L_{ij}$. luego repetir el paso 6.

Si $L_{ij} < U$, y la solución no define una ruta para TSP, colocar TSP_{ij} en la lista y luego repetir el paso 6.

En ambas versiones, el procedimiento puede ser representado mediante un árbol (árbol de búsqueda o árbol de ramificación y acotación) cuyos nodos corresponden a los sub-problemas generados, con la raíz correspondiente al problema original, y los nodos sucesores de un nodo dado *i* asociado con TSP_i correspondientes a los sub-problemas $TSP_{i1},...,TSP_{iq}$, definidos mediante alguna regla de ramificación.

La diferencia entre uno y otro algoritmo, basado en el método de RA, radica

fundamentalmente en:

- La relajación del TSP_i
- La regla de ramificación
- La regla de elección de un subproblema, y
- La heurística utilizada para calcular la cota superior

Respecto a la elección de un sub-problema, puede utilizarse la búsqueda en profundidad, es decir, eligiendo siempre el nodo de más reciente creación, o la búsqueda en amplitud es decir eligiendo un nodo de las generaciones primeras.

El punto básico en los algoritmos de RA, lo constituyen las relajaciones y las reglas de ramificación. Existen muchas relajaciones utilizadas, y a su vez dentro de cada relajación varias reglas de ramificación. Cada una de ellas determinan algoritmos diferentes. Una revisión de las relajaciones más importantes con sus respectivas reglas de ramificación se puede encontrar en [6,111], las mismas que revisamos a continuación.

Históricamente la primera relajación que se ha utilizado, de gran importancia incluso en la actualidad, es la relajación a *un problema de asignación* (PA), introducida por Eastman. Esta relajación de TSP, obtenida a partir de la formulación como un problema de programación lineal, consiste en un problema de asignación, dado por el problema (3.39), pero excluyendo la restricción (3.38), es decir sin considerar la restricción que evita la obtención de las sub-rutas, y se nota por PA(TSP). Asociada a esta relajación se han utilizado varias reglas de ramificación, siendo los más destacados: la regla propuesta por Eastman, la propuesta por Little, Murty, Sweeney y Karel, la propuesta por Murty, la propuesta por Bellmore y Malone y la propuesta por Garfinkel.

La segunda relajación en aparecer fue la relajación como *problema de 1-árbol con función objetivo lagrangiana* propuesta por Held y Karp para resolver el TSP simétrico. Esta relajación, obtenida a partir de la representación del TSP como un grafo no dirigido, consiste en encontrar un sub-grafo generador conexo H del grafo G, con aristas que minimizan la función objetivo (3.40). Este sub-grafo H se denomina 1-árbol que minimiza (3.40). Para facilitar la obtención del tal 1-árbol, se combina la función objetivo (3.40) y las restricciones

(3.41) mediante multiplicadores de lagrange, razón por la cual también se denomina *relajación lagrangiana.*

Balas y Christofides [5] en 1981 propusieron el uso de una relajación obtenida mediante la unión de la relajación lagrangiana y la relajación PA(TSP), para resolver el TSP asimétrico. Esta unión consistiría en considerar el problema de asignación, pero con la función objetivo dada por el lagrangiano.

3.3.2.1 El algoritmo LMSK

Denominamos *algoritmo LMSK* al algoritmo que utiliza la relajación como problema de asignación, y la regla de ramificación propuesta por Little, Murty, Sweeney y Karel [120]. La idea de este algoritmo es muy natural y simple. En principio el TSP inicial se resuelve considerandolo como un simple problema de asignación, usando el método Húngaro; esta solución obtenida, si no posee sub-rutas, constituye la solución óptima del TSP. En el caso contrario, utilizando la respectiva regla de ramificación, se elige una variable x_{ij} y se obtienen dos sub-problemas, fijando el valor 1 y 0 respectivamente, para la variable elegida. En este procedimiento se genera un árbol de decisión binario, y el algoritmo se detiene cuando en todas las ramas se han obtenido soluciones sin sub-rutas o cuando hay un indicador de que en las ramas con sub-rutas, no existe la posibilidad de mejorar el resultado existente.

Este algoritmo utiliza la versión 1 del procedimiento general del método de ramificación y acotación para TSP. En el *paso* 2, la regla de elección consiste en elegir el sub-problema TSP_i de más reciente creación eligiendo primero aquel sub-problema en la que el valor para la última variable fijada es 1. Los pasos opcionales 4 y 5 no los utiliza. Y en el paso 6 utiliza la siguiente regla de ramificación:

En un sub-problema TSP_k (nodo k) como consecuencia de las ramificaciones anteriores existen variables x_{ij} con valores fijados (1 o 0), o dicho en término de grafo existen aristas (i,j) incluidas o excluidas en la ruta. Notamos I al conjunto de las aristas incluidas, E al conjunto de las aristas excluidas. Entonces, la relajación de TSP_k, es decir el sub-problema

de asignación en el nodo k, puede ser escrito como:

$$\begin{aligned} \min \quad & \sum_{(i,j)\in I} d_{ij} + \sum_{i\in S}\sum_{j\in T} d'_{ij}\, x_{ij} \\ \text{s.a.} \quad & \sum_{i\in S} x_{ij} = 1, \quad \forall j \in T, \\ & \sum_{j\in T} x_{ij} = 1, \quad \forall i \in S, \\ & x_{ij} \in \{0,1\}, \quad \forall (i,j), \quad i \in S, \quad j \in T \end{aligned} \tag{3.45}$$

donde:
$$S=\{i/(i,j)\notin I \ \forall j\}, \quad T=\{j/(i,j)\notin I \ \forall i\} \quad \text{y} \quad d'_{ij}=\begin{cases} d_{ij} & (i,j)\notin E \\ \infty & (i,j)\in E \end{cases}$$

además d_{ij} son los coeficientes de la matriz de distancias reducidas del nodo anterior, es decir, la matriz que queda después de obtener la asignación óptima en el nodo anterior.

Este sub-problema se resuelve usando el método Húngaro, si la solución obtenida tiene sub-rutas se procederá a la ramificación. Una regla de ramificación es elegir una variable x_{rs} donde $r \in$ S y $s \in$ T, y crear dos nodos asignando el valor 1 o 0 respectivamente. Little, Murty, Sweeney y Karel [120] sugieren seleccionar, para ramificar, una variable x_{rs} que tenga el máximo potencial de incremento en la función objetivo del sub-problema, cuando fijamos en x_{rs} el valor cero. Para hacer esto, sea

$$\{\bar{d}_{ij}\} \quad i \in S, \quad j \in T$$

el costo reducido de la solución óptima del sub-problema. Entonces para cada arco (i,j), $i\in$S, $j\in$T con costo reducido 0, calculamos:

$$p_{ij} = min\{\bar{d}_{ih} / h \in T-\{j\}\} + min\{\bar{d}_{hj} / h \in S-\{i\}\} \tag{3.46}$$

cantidad mínima en la que se incrementará el valor de la asignación óptima del sub-problema, si la variable x_{ij} es fijado a cero. Por tanto podemos escoger x_{rs} tal que:

$$p_{rs} = max\{p_{ij} / i \in S, j \in T, \quad \bar{d}_{ij} = 0\} \tag{3.47}$$

Una vez escogida la variable de ramificación x_{rs}, los dos nuevos nodos se obtienen haciendo x_{rs} =1 y x_{rs} =0 respectivamente, es decir en el primer nodo nuevo, el conjunto I tiene como nuevo elemento al arco (r,s), y en el segundo nuevo nodo el conjunto E tiene al arco (r,s), como un nuevo elemento.

Resumen del algoritmo LMSK

Paso 1: [Inicialización] Hacer U=∞ (mejor cota y valor actual) y L={TSP} (lista de sub-problemas).

Paso 2: [Selección de un sub-problema] Si L=ϕ detener el proceso, pues la ruta asociada con U es óptima (si U=∞ el TSP no tiene solución).
Si L≠ϕ, elegir el sub-problema TSP_i de más reciente creación, y eliminarlo de la lista L. Ir al paso 3.

Paso 3: [Determinación de una cota superior] Resolver mediante el método Húngaro el AP(TSP_i). Sea Z_i el valor obtenido:
Si $Z_i \geq U$, regresar al paso 2.
Si $Z_i < U$ y la correspondiente solución es una ruta para TSP (es decir no existen subrutas) entonces hacer $U=Z_i$.
Si $Z_i < U$, y la correspondiente solución no define una ruta para TSP (es decir si existen sub-rutas) ir al paso 4.

Paso 4: [Ramificación] Elegir x_{rs} de acuerdo a (3.47), y generar dos nuevos sub-problemas TSP_{i1} y TSP_{i2}, fijando $x_{rs}=0$ y $x_{rs}=1$ respectivamente. Hacer $L=L\cup\{ TSP_{i1}, TSP_{i2}\}$. Ir al paso 2.

Observación: Resaltamos el criterio de parada de este algoritmo, que es la condición de que la lista L sea vacía, es decir no queda ningún sub-problema por resolver.

3.3.3 Cotas del valor optimo de un TSP

En general todo algoritmo aproximado o heurística proporciona una cota superior del valor óptimo de un TSP. Una revisión de estos tipos de algoritmos podemos encontrarla en

[73, 97, 111, 131, 198].

3.3.3.1 Cota superior

La cota superior para el valor óptimo del TSP que utilizaremos en este trabajo será obtenida mediante la heurística denominada *Procedimiento de Construcción de una Ruta.* Es una heurística que se inicia con pocas ciudades, pero va adicionando una a una las ciudades restantes en el recorrido. El orden y la forma en que adiciona cada ciudad se basa en algún criterio que habrá que fijar, y que le confiere una marcada naturaleza voraz. Por lo tanto en esta heurística se distinguen tres componentes:

- La elección de una sub-ruta inicial.
- El criterio de selección, y
- El criterio de inserción.

Existen diferentes heurísticas con estas componentes pero que se diferencian entre si por el procedimiento utilizado en cada una de ellas. Aquí, debido a su eficiencia y simplicidad, hemos elegido la heurística denominada *Procedimiento de la Inserción Arbitraria,* propuesto por Rosenkrantz, Stearns y Lewis [162]

El Procedimiento de la Inserción Arbitraria se inicia con un sub-grafo consistente en una sola ciudad arbitraria i (es decir el recorrido $i \rightarrow i$), luego selecciona la ciudad más cercana k y genera una sub-ruta (i, k) (es decir $i \rightarrow k \rightarrow i$). Para insertar las demás ciudades, selecciona una siguiente ciudad k arbitrariamente e inserta entre las ciudades i y j con el menor $d_{ik} + d_{kj} - d_{ij}$. Para precisar mejor el procedimiento, hacemos un pequeño reajuste en el paso inicial, eligiendo la ciudad 1 en lugar de una ciudad arbitraria i, y en la selección, seleccionando en forma ordenada las ciudades 2,3,...,n sucesivamente, quedando finalmente de la siguiente manera:

Procedimiento de la inserción arbitraria:

Paso 1: [Inicio] Inicialice la sub-ruta $1 \rightarrow i \rightarrow 1$, con i que minimiza $d_{1i} + d_{i1}$.

Paso 2: [Selección] Seleccione la ciudad $k+1$ ($k+1 \neq i$) siendo k la última ciudad insertada.

Paso 3: [Inserción] Encontrar el arco $\{i,j\}$ en la sub-ruta el cual minimiza $d_{ik}+d_{kj}-d_{ij}$. Inserte k entre i y j.

3.3.3.2 Cota inferior

Una cota inferior para el valor óptimo del TSP es el valor óptimo de su relajación. Particularmente el valor óptimo de PA(TSP) constituye una cota inferior. Más aún a partir de la matriz reducida, producida por la asignación óptima, se puede obtener una mejor cota inferior, si esta asignación no proporciona una ruta de TSP (es decir si no existen sub-rutas). En este último caso, la función objetivo de TSP se puede escribir en la forma:

$$z = z_A + \sum_{i=1}^{n}\sum_{j=1}^{n} \bar{d}_{ij} x_{ij}$$

donde: z_A es el valor de la asignación óptima, y
$\bar{d}_{ij}$ es la distancia reducida producida por la asignación óptima.

Ahora, definamos el grafo G_0 que incluye todos los n nodos y sólo los arcos (i,j) correspondientes a la distancia reducida cero, es decir, $\bar{d}_{ij}=0$. A su vez, en el grafo G_0 definimos como *componente* a un conjunto de nodos K si no existe ningún arco (i,j), $i\in K$ y $j\in\bar{K}$ o $i\in\bar{K}$ y $j\in K$, $\bar{K}=\{1,2,...,n\}-K$ y ningún subconjunto propio de K satisface esta condición.

Sean $K_1,...,K_r$ todas las componentes de G_0, entonces puede demostrarse que

$$L = z_A + \sum_{l=1}^{r} \underset{i\in K_l, j\in\bar{K}_l}{min} \{\bar{d}_{ij}\} \qquad (3.48)$$

es una cota inferior de TSP [175].

3.3.4 Criterios de parada difusos en algoritmos del TSP

Solo hay algoritmos exactos que resuelven el TSP para casos especiales o para un número reducido de ciudades, como podemos ver en [27, 72]. Para los casos generales se han desarrollado algoritmos aproximados o heurísticas de tipo clásico, como podemos ver en [73,97] y heurísticas que utilizan procedimientos basados en las leyes naturales, como son los algoritmos genéticos, recocido simulado etc. [111, 131, 198].

La propuesta que hicimos para los algoritmos de la PL y PM, consistente en la introducción de criterios de parada difusos, también la extendemos al TSP para obtener nuevos algoritmos de tipo aproximado, pero sobre la base de los algoritmos exactos existentes en este caso..

Cuando incluimos los criterios de parada difusos en los algoritmos exactos, lo que hacemos es flexibilizar el algoritmo de tal manera que puede ser manipulado fácilmente por el decisor, es decir, si el tiempo de ejecución para obtener la solución óptima es muy prolongado, el decisor puede considerar aceptable una solución cercana a la óptima, pero obtenida en un tiempo razonable.

Para esto, consideramos que el valor de la solución óptima del TSP no solo es una incógnita, sino también un valor impreciso, debido a que valores cercano a la óptima puede ser admitidos por el decisor como si lo fueran, es decir, es un conjunto difuso con soporte determinado por una cota inferior L_0 y una cota superior U_0, y mediante una función de pertenencia:

$$\mu(z) = \begin{cases} 1 & \text{si } z < L_0 \\ f(z) & \text{si } L_0 \le z \le U_0 \\ 0 & \text{si } z > U_0 \end{cases} \tag{3.49}$$

donde: $f(z) \in [0,1] \ \forall z \in [L_0, U_0]$, es una función continua no creciente.

Esta función de pertenencia indica que si el valor z de una ruta TSP es superior a U_0, no es admitido por el decisor. Un valor menor que L_0 sería una buena solución, y los valores entre L_0 y U_0 son aceptables, pero el grado de aceptación se incrementa, a medida que

disminuye, hasta alcanzar el mayor grado de aceptación cuando alcanza el valor L_0.

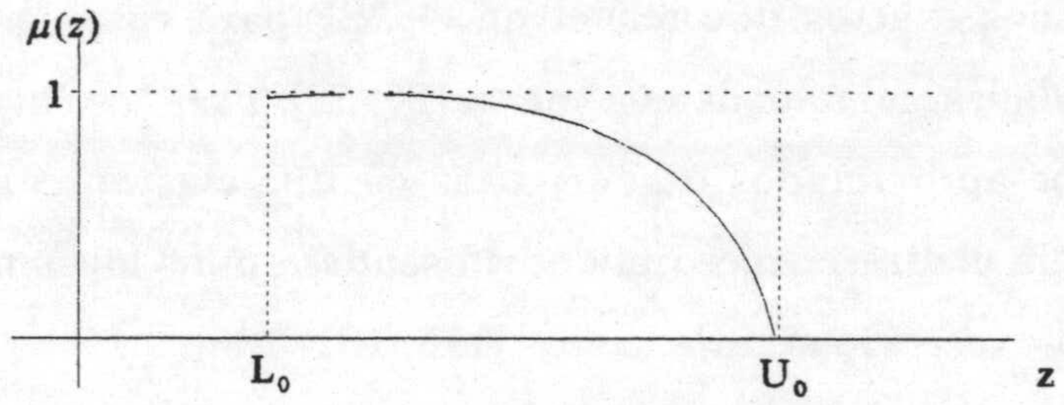


Figura 3.3: Función de pertenencia cóncava

Si el decisor considera aceptable una solución no óptima, con un grado de pertenencia no menor que α ($0<\alpha<1$), una de las condiciones de parada estaría dada por:

$$\mu(z) \geq \alpha \quad o \quad z \leq f^{-1}(\alpha) \tag{3.50}$$

Para que esta condición proporcione los resultados esperados por el decisor, los valores L_0, U_0 y la función f en la definición del conjunto difuso deben ser adecuados. Unos valores inapropiadas de L_0 y U_0 pueden conducir a soluciones con grandes errores. Así mismo una función inadecuada puede anular la flexibilización.

En la definición de la función de pertenencia, por las mismas razones expresadas en la sección 3.2.4 correspondiente al PM, es recomendable utilizar una función cóncava f, como el de la Figura 3.3. Igualmente, un ejemplo de estos tipos de funciones es:

$$f(z) = \sqrt[n]{\frac{U_0 - z}{U_0 - L_0}} \tag{3.51}$$

y las cotas L_0 y U_0 pueden ser determinadas mediante heurísticas apropiadas y eficientes.

Si flexibilizamos los algoritmos de RA del TSP, una buena cota inferior está dada por el valor óptimo de una relajación. Mientras que una cota superior apropiada puede obtenerse como en la sección 3.3.3.1.

3.3.4.1 El criterio de parada difuso en el algoritmo LMSK

Como ilustración, vamos a introducir un criterio de parada difuso en el algoritmo LMSK. Para definir la función de pertenencia, en el paso inicial se debe resolver mediante el método Húngaro el PA(TSP). Si la asignación óptima no constituye una ruta para el TSP, calculamos L_0 como en la sección 3.3.3.2 y U_0, como en la sección 3.3.3.1. También en el mismo paso inicial, el decisor, deberá elegir el grado de aceptación mínima α. Finalmente en el paso 2 debe incluirse la condición de parada difusa (3.50). Así tenemos el siguiente algoritmo:

Paso 1: [Inicialización] Hacer $L=\phi$. Resolver mediante el método Húngaro PA(TSP).
Si la asignación óptima es una ruta de TSP ir al paso 2.
En caso contrario, ir al paso 1'.

Paso 1': Hallar L_0 según (3.48) y U_0 como en la sección 3.3.3.1, luego hacer $U=U_0$ (mejor cota actual) e ir al paso 1";

Paso 1": Dar el valor α ($0<\alpha\leq 1$).
Si $0<\alpha<1$ hacer $z_0 = f^{-1}(\alpha)$ (cota para la solución aceptable, donde f es como en (3.49)) y si $L\neq\phi$ ir al paso 2. En caso contrario ir al paso 4.
Si $\alpha=1$ (no quiere mejorar la solución aceptable), hacer $L=\phi$ e ir al paso 2.

Paso 2: [Selección de un sub-problema] Si $L = \phi$ o $U\leq z_0$ detener el proceso, pues la ruta asociada con U es aceptable; si $L\neq\phi$ ir al paso 1". En caso contrario salir.
Si $L\neq\phi$ y $U > z_0$, elegir el sub-problema TSP_i de más reciente creación, eliminarlo de la lista L e ir al paso 3.

Paso 3: [Determinación de una cota superior] Resolver mediante el método Húngaro el PA(TSP_i). Sea Z_i el valor obtenido:
Si $Z_i \geq U$, regresar al paso 2.
Si $Z_i < U$ y la correspondiente solución es una ruta para TSP (es decir no existe sub-rutas) entonces hacer $U \leftarrow Z_i$.
Si $Z_i < U$ y la correspondiente solución no define una ruta para TSP (es decir si existen sub-rutas) ir al paso 4.

Paso 4: [Ramificación] Elegir x_{rs} de acuerdo a (3.47) y generar dos nuevos sub-problemas TSP_{i1} y TSP_{i2} fijando $x_{rs}=0$ y $x_{rs}=1$ respectivamente, y hacer $L \leftarrow L \cup \{ TSP_{i1}, TSP_{i2} \}$, luego ir al paso 2.

Al introducir el criterio de parada difuso se ha flexibilizado el algoritmo de tal manera que el decisor puede controlar las iteraciones, pues en el paso 1" puede introducir valores pequeños para α e ir incrementando si desea mejorar la solución aceptable. Para esto naturalmente tendrá en cuenta el tiempo utilizado en la obtención de las anteriores soluciones aceptables.

3.3.4.2 Ejemplo numérico

Consideremos el TSP de 10 ciudades, con matriz de distancias siguiente:

$$(d_{ij}) = \begin{bmatrix} - & 1 & 62 & 56 & 54 & 27 & 30 & 27 & 55 & 60 \\ 90 & - & 66 & 77 & 52 & 98 & 12 & 55 & 7 & 64 \\ 30 & 41 & - & 60 & 59 & 17 & 72 & 82 & 76 & 21 \\ 57 & 33 & 33 & - & 64 & 78 & 62 & 24 & 70 & 72 \\ 95 & 32 & 69 & 74 & - & 97 & 94 & 92 & 96 & 55 \\ 29 & 25 & 40 & 61 & 25 & - & 27 & 81 & 57 & 94 \\ 98 & 52 & 8 & 11 & 89 & 61 & - & 55 & 91 & 37 \\ 52 & 50 & 90 & 33 & 64 & 86 & 37 & - & 91 & 88 \\ 45 & 83 & 31 & 79 & 70 & 22 & 46 & 18 & - & 91 \\ 96 & 62 & 88 & 9 & 2 & 67 & 64 & 43 & 85 & - \end{bmatrix} \quad (3.52)$$

Los elementos de la diagonal de la matriz de distancias, así como las distancias de arcos excluidos durante las iteraciones, los consideramos con valor M=10x(max d_{ij}), es decir d_{ii}=M y d_{ij}=M si el arco (i,j) es excluido de una posible ruta.

Resolviendo el problema mediante el algoritmo exacto LMSK, se obtiene la ruta óptima

$$1 \rightarrow 2 \rightarrow 9 \rightarrow 6 \rightarrow 5 \rightarrow 10 \rightarrow 4 \rightarrow 8 \rightarrow 7 \rightarrow 3 \rightarrow 1$$

con distancia total z=218, después de resolver en total 15 sub-problemas incluido el problema original. En la Figura 3.4 se presenta el árbol de ramificación y acotación correspondiente, en donde la enumeración de los nodos corresponde al orden en que el algoritmo LMSK resuelve los sub-problemas.

Utilizando el algoritmo LMSK con criterio de parada difuso, en el que la función *f* está dada por (3.51), con n=2, se obtiene L_0=208, U $\bar{0}$308 y las soluciones aceptables para distintos valores de α se muestran en la Tabla 3.5.

α	$z_0=f^{-1}(\alpha)$	Ruta aceptable	Valor aceptable	No. Sub-probl. resueltos (Itera.)
0.5	283	1-7-3-10-5-2-9-8-4-6-1	258	8
0.8	244	1-8-7-4-3-19-5-2-9-6-1	221	10
0.94	219.64	1-2-9-6-5-10-4-8-7-3-1	218	14

Tabla 3.5: Soluciones aceptables para el ejemplo según el algoritmo LMSK con criterio de parada difuso.

Se observa que para α=0.8, la solución aceptable se obtiene resolviendo sólo las 2/3 partes del total de los sub-problemas que el algoritmo clásico tiene que resolver. Sin embargo el valor aceptable que se obtiene es muy cercano al óptimo. Por lo tanto es evidente que la proporción del tiempo que se ahorra es muy superior en comparación con el diferencial del valor aceptable con valor óptimo. Además para α=0.94 se obtiene una solución aceptable que es exacta, en menos iteraciones que con el algoritmo original clásico. Estas ventajas se hacen más evidentes cuanto mayor es el número de ciudades en el TSP.

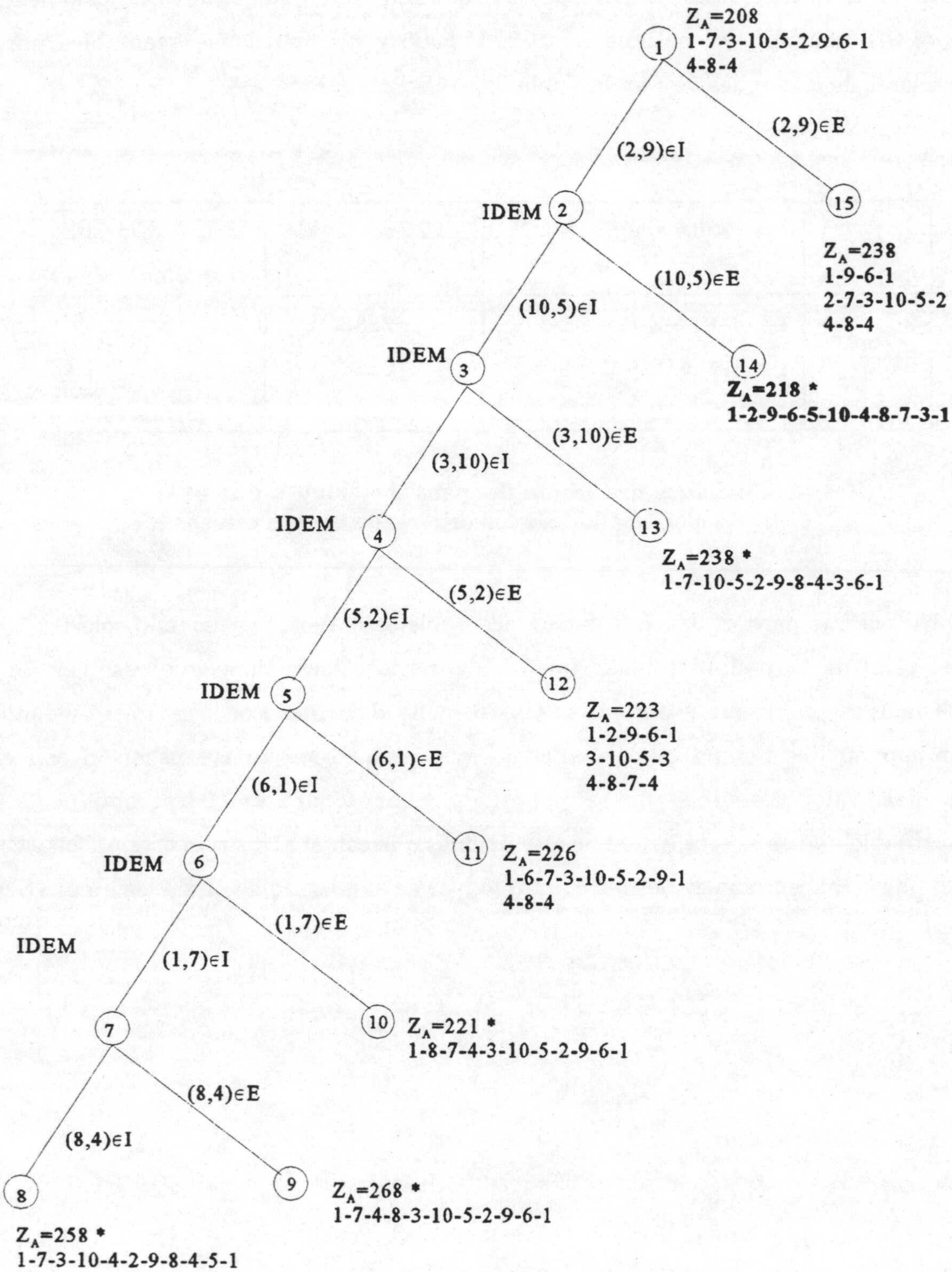


Figura 3.4: Arbol de ramificación y acotamiento el ejemplo. * indica que son rutas del TSP, y la negrita indica la ruta óptima.

COMENTARIOS FINALES

Con la constante variación de las características del mundo empresarial, nunca una base de modelos de un sistema de soporte de decisión (SSD) es completa, lo que se evidencia con la incesante aparición de problemas reales para los cuales en el SGBM no existen modelos con sus correspondientes métodos de solución apropiados o bien los existentes presentan limitaciones. Ante esta situación se presentarían dos opciones a seguir (no excluyentes o mejor dicho complementarias): una, la *actualización* permanente del SGBM y otra, el *diseño de nuevos modelos y métodos de resolución o el reajuste de los existentes* para resolver los nuevos problemas.

En lo que respecta a este trabajo, como se expuso en la introducción, nos ha preocupado de entre los problemas de la asignación de recursos limitados y del control de acciones cambiantes con el tiempo, los siguientes aspectos:

1) La toma de decisiones en ambiente semi-estructurado, es decir, en realidades donde los datos son imprecisos; y
2) Las limitaciones que presentan los modelos y métodos existentes en un ambiente estructurado para casos de grandes dimensiones.

Para el primer aspecto, correspondiente a situaciones semi-estructuradas, hemos optado por el camino de la *actualización del SGBM* recogiendo en un solo documento los principales modelos y métodos de la programación lineal difusa, junto con sus extensiones más importantes y sus aplicaciones. Todo ello ha ocupado la primera parte de la presente Memoria.

Al abordar el segundo aspecto, referente a las limitaciones que presentan los modelos existentes en casos complejos, se ha optado por proponer nuevas alternativas basadas en propuestas y teorías también nuevas. Esto ha ocupado la segunda parte de esta Memoria, donde nos hemos dedicado concretamente a las limitaciones, en problemas NP-completos, de los algoritmos exactos; eficientes para ejemplares pequeños o con

características particulares pero ineficientes en la generalidad para grandes dimensiones. Las alternativas que hemos propuesto se basan en la propuesta de Herrera y Verdegay [81] sugiriendo el uso de las reglas de control difusas en tales algoritmos, y que nos han conducido a formular las conclusiones y líneas futuras siguientes.

CONCLUSIONES

I. La introducción de las reglas de control difusas en los algoritmos exactos los transforma en flexibles y dinámicos, susceptibles de ser manipulados por el decisor para adaptarlos a situaciones y criterios concretos, de interés sólo para el decisor.

II. La etapa que caracteriza la rigidez o flexibilidad de un algoritmo es aquélla en que se decide continuar o terminar las iteraciones, por lo tanto la flexibilización se efectúa en los criterios de parada introduciendo las reglas difusas, transformándose éstos en criterios de parada difusos.

III. Los criterios de parada difusos están asociados con conjuntos difusos y éstos, definidos a su vez mediante funciones de pertenencia. Estas funciones de pertenencia deben ser no lineales, convexas o cóncavas, de acuerdo a las particularidades de los problemas, para obtener los mejores resultados.

IV. El uso de los criterios de parada difusos en los algoritmos exactos permite obtener nuevos algoritmos de tipo heurístico que proporcionan soluciones aceptables (cercanas al óptimo) en tiempos de ejecución razonables.

V. En algunas situaciones, los criterios de parada difusos pueden permitir obtener soluciones óptimas en menor número de iteraciones, y en consecuencia, en menor tiempo de ejecución que el correspondiente algoritmo exacto.

Adicionalmente, al introducir los criterios de parada difusos en algoritmos de problemas con características más específicas se ha llegado a las siguientes conclusiones:

A. <u>En los algoritmos de la programación Lineal</u>:

A.1 En el algoritmo simplex:

Dado el criterio de parada: $\delta_j \geq 0 \quad \forall j$, con expresión lingüística "δ_j no negativa" . Herrera y Verdegay proponen flexibilizarlo con el criterio difuso de expresión lingüística "δ_j poco negativo" que conduce al criterio de parada difuso:

$$\delta_j \geq h_j^{-1}(\alpha) \ , \ \alpha \in (0,1], \ \forall j \qquad (*)$$

donde h_j es la función de pertenencia lineal y α el nivel de tolerancia.

En este trabajo se propone:

- En lugar de (*) la condición:

$$\delta_j / u \geq (h_j^{-1}(\alpha)) / m \ , \ \alpha \in (0,1], \ \forall j$$

donde u es el número de la iteración actual y m el número de restricciones. Esta condición se adecua mejor a situaciones más variadas.

- Usar una función no lineal y convexa h, porque proporciona mejores aproximaciones con niveles de tolerancia razonables.

A.2 En el algoritmo de Karmarkar:

Este algoritmo resuelve la forma estándar de Karmarkar que consiste en minimizar una función objetivo no negativo con mínimo cero. El número de iteraciones de este algoritmo es fijo y depende de la dimensión del problema, aunque puede finalizar el proceso cuando alcanza el mínimo, es decir, cuando cx=0.

La flexibilización que se propone, al introducir una regla de parada difusa, conduce al criterio difuso:

$$cx \geq g^{-1}(\alpha)$$

con g función de pertenencia:

- Convexa, si se presume que el mínimo es cero.
- Cóncava si se presume que el mínimo es positivo.

El algoritmo de Karmarkar incluye en su proceso indicadores para determinar si el mínimo es cero o positivo.

A.3 En algoritmos modificados de Karmarkar:

Tanto en el algoritmo de modificación que proponen Vanderbei y otros, como en el que propone Barnes, los criterios de parada difusos que se derivan del uso de funciones de pertenencia convexa proporcionan mejores soluciones aceptables con niveles de tolerancia moderados.

A.4 En los algoritmos de puntos interiores

En los algoritmos de puntos interiores, algoritmo primal, algoritmo dual y algoritmo primal-dual, que tienen los mismos criterios de parada, el uso de una función no lineal convexa en el criterio de parada difusa proporciona mejores soluciones aceptables para niveles de tolerancia moderados.

B. <u>En los algoritmos del problema de la mochila</u>:

Los algoritmos exactos más importantes del problema de la mochila son los basados en los métodos de ramificación y acotación, por un lado, y los basados en la programación dinámica, por otro lado. En algunos de los más eficientes de estos tipos de algoritmos hemos introducido criterios de parada difusos, en los que ha quedado verificado la ventaja de utilizar funciones de pertenencia no lineales cóncavas. De esta manera se han obtenido algoritmos de aproximación que son más eficientes que el algoritmo aproximado de Sahni, haciéndose más significativa la ventaja cuando el número de ítems se incrementa.

C. <u>En los algoritmos del problema del viajante de comercio</u>:

En el problema del viajante de comercio, un algoritmo exacto de mayor importancia por su aplicabilidad a situaciones muy generales proviene del uso del método de ramificación y acotación, razón por la cual hemos dedicado nuestra atención en este algoritmo para introducir los criterios de parada difusos.

También aquí se ha verificado que la utilización de las funciones no lineales cóncavas en la definición de los criterios de parada difusos son de aplicabilidad general y proporcionan mejores resultados.

LÍNEAS FUTURAS:

El uso de reglas difusas ha tenido y tiene significativa importancia en muchos aspectos de la Inteligencia Artificial, y posiblemente ello alentó a Herrera y Verdegay a proponer el uso de las reglas difusas en los algoritmos exactos para ensanchar su aplicabilidad. Pese a que desde la primera propuesta han transcurrido casi cuatro años, la utilización de las reglas difusas en los diferentes algoritmos exactos no ha hecho más que empezar. Los resultados que se han obtenido en este trabajo refuerzan las posibilidades de tener muchos éxitos cuando se utilicen en distintos tipos de algoritmos exactos, no sólo en problemas de tipo NP-completos sino también en aquéllos en los que existen algoritmos exactos con tiempos de ejecución teóricamente polinomiales pero que en la práctica requieren tiempos prolongados.

Por lo tanto, las líneas futuras en este contexto están abiertas en todos los ámbitos, como el de la programación no lineal por citar alguno.

En el contexto de la programación lineal en la que se ha trabajado, resta sin embargo un largo camino por recorrer, en particular, y dentro del problema de la mochila, los algoritmos de problemas multiobjetivos, de múltiple elección, etc, más largo si cabe por lo que respecta al problema del viajante de comercio.

En cualquier caso, las expectativas de aplicación y desarrollo de los SDD en la sociedad de la información, son tan importantes que la continuidad de este trabajo, tanto desde el punto de vista de la disponibilidad del autor, como del de la necesidad en el entorno social donde se ha de desenvolver, está absolutamente asegurada a corto y medio plazo.

BIBLIOGRAFÍA

[1] Abboud, N.J., M. Sakawa y M. Inuiguchi, A fuzzy programming approach to multiobjective multidimensional 0-1 knapsack problems, *Fuzzy Sets and Systems* 86 (1997) 1-14.

[2] Aho, A.V., J.E. Hopcroft y J.D. Ullman, The design and Analysis of Computer Algorithms, Addison Wesley P.C., California-USA. 1974.

[3] Arbel, Ami, *Exploring Interior-Point Linear Programming: Algorithms and Software*, Mit Pres, London, 1993.

[4] Balas, E., A note on the branch-and-bound principle, *Operations Research* 16(2)(1968)442-445.

[5] Balas, E. Y N. Christofides, A restricted Lagrangean approach to the traveling salesman problem, *Mathematical Programming* 21 (1981) 19-46.

[6] Balas, E. Y P. Toth, Branch and bound methods, Cap. 10 de [113], (1985) 361-402.

[7] Balinski, M., Integer programming:methods, uses, computation, *Management Science* 12 (2)(1965) 253-313.

[8] Baptistella, L.F.B. y A. Ollero, Fuzzy methodologies for interactive multicriteria optimization, *IEEE Transaction Systems Man Cybernet.* 10 (1980) 355-365.

[9] Barnes, E.R., A variation on Karmarkar's algorithm for solving linear programming problems, *Mathematical Programming* 36 (1986) 174-182.

[10] Bayer, D.A. y J.C. Lagarias, Karmarkar's linear programming algorithm and Newton´s method, *Mathematical programming* 50 (1991) 291-330.

[11] Bazaraa, M.s. y C.M. Shetty, *Nonlinear Programming, Theory and Algorithms*, John Wiley and sons 1979.

[12] Bellman, R., Some aplications of the theory of dynamic programming: A review, *Operations Research* 2 (2) (1954) 275-288.

[13] Bellman, R.E. y Zadeh, L.A. Decision-making in a fuzzy environmement, *Management Science* 17 (1970) B141-B164.

[14] Benoit, J., An extension to possibilistic linear programming, *Fuzzy Sets and Systems* 64 (1994) 195-206.

[15] Benson, H. P., Multiple objetive linear programming with parametric criteria coefficients. *Man. Sci.* 31, 4 (1985) 694-706.

[16] Berloux P. y P. Bizard, *Algoritms: the construction, proof and analysis of programs*, John Wiley and sons 1987.

[17] Bitran, G. R.; Linear multiple objective problems with interval coeficients, *Man. Sci.* 26 (1980) 7, 694-706.

[18] Bit, A.K., M.P. Biswal y S.S. Alam, Fuzzy programming approach to multicriteria decision making transportation problem, *Fuzzy Sets and Systems* 50 (1992) 135-141.

[19] Bit, A.K., M.P. Biswal y S.S. Alam, Fuzzy programming approach to multiobjective solid transportation problcm, *Fuzzy Sets and Systems* 57 (1993) 183-194.

[20] Bland, R.G. y D.F. Shallcross, Large traveling salesman problems arising experiments in X-ray crystallography; a preliminary report on computation, *Operations Research Letters* 8 (1989) 125-128.

[21] Braasard G. Y P. Bratley, *Algoritmica:Concepción y Análisis*, Masson (ed. Española), Barcelona 1990.

[22] Brassard G. Y P. Bratley, *Fundamentos de Algorítmia*, Prentice Hall, Madrid, 1ª reimpresión 1998 (Primera edición 1997).

[23] Buckley , J. J. Fuzzy programmming and the pareto optimal set, *Fuzzy Sets and Systems* 10(1983) 57-63.

[24] Buckley, J.J., Possibilistic linear programming with triangular fuzzy numbers, *Fuzzy Sets and Systems* 26 (1988) 135-138.

[25] Buckley, J.J., Solving possibilistic linear programming problems, *Fuzzy Sets and Systems* 31 (1989) 329-341.

[26] Buckley, J.J., Multiobjective possibilistic linear programming, *Fuzzy Sets and Systems* 35 (1990)23-28.

[27] Burkard, R.E., V.G. Deineko, R. Van Dal, J.A.A. Van Der Veen y G.J. Woeginger, Well-solvable special cases of the traveling salesman problem: Λ survey, *Siam Review*, vol. 40 No.3 (1998) 496-546.

[28] Cadenas, J.M. y J.L. Verdegay, Using ranking functions in multiobjective fuzzy linear programming, por aparecer en *Fuzzy Sets and Systems.*

[29] Cadenas, J.M. y J.L. Verdegay, PROBO: an interactive system in fuzzy linear programming, *Fuzzy Sets and Systems* 76 (1995) 319-332.

[30] Carlsson, Ch. y P. Korhonen; A parametric approach to fuzzy linear programming. *Fuzzy Sets and Systems* 20(1986) 17-30.

[31] Castro, J.L., F. Herrera y J.L. Verdegay, Solving linear boolean programming problems with imprecise costs, *Proceedings of the IEEE International Conference on Fuzzy Systems* 1992, San Diego(1992) 1025-1032.

[32] Campos, L., Fuzzy linear programming models to solve fuzzy matrix games, *Fuzzy Sets and Systems* 32 (1989) 275-289.

[33] Campos, L. y J.L. Verdegay, Linear programming problems and ranking of fuzzy numbers, *Fuzzy Sets and Systems* 32 (1989) 1-11.

[34] Chalam, G.A. , Fuzzy goal programming (FGP) approach to a stochastic transportatión problcm undcr budgctary constraint, *Fuzzy Sets and Systems* 66 (1994) 293-299.

[35] Chanas S., The use of parametric programming in FLP, *Fuzzy Set and Systems* 11 (1983) 243-251.

[36] Chanas, S., Fuzzy programming in multiobjetive linear programming - A parametric aproach, *Fuzzy Sets and Systems* 29 (1989)303-313.

[37] Chanas, S., M. Delgado, J.L. Verdegay y M.A. Vila, Interval and fuzzy extensions of classical transportation problcms, *Transportation Planing and Technology* 17 (1993) 203-218.

[38] Chanas, S., W. Kolodziejczyk y A. Machay, A fuzzy approach to the transportation problem, *Fuzzy Sets and Systems* 13 (1984) 221-221.

[39] Chanas, S. y D. Kuchta, Fuzzy integer transportation problem, *Fuzzy Sets and Systems* 98 (1998) 291-298.

[40] Chang, S-Y., E.D. Jr. Brill y L.D. Hopkins, Modelling to generate alternatives: A fuzzy approach, *Fuzzy Sets and Systems* 9 (1983) 137-151.

[41] Chang, N-B, Y.L. Chen y S.F. Wang , A fuzzy interval multiobjective mixed integer programming approach for the optimal planing of solid waste management systems, *Fuzzy Sets and Systems* 89 (1997) 35-60.

[42] Charnes, A. Y W.W. Coper, *Manegement Models and Industrial Applications of Linear Programming*, John Wiley, New York, 1961.

[43] Chen, H.K. A note on a fuzzy goal programming algorithm by Tiwari, Dharmar and Rao, *Fuzzy Sets and Systems* 62 (1994)287-290.

[44] Christofides, N., Vehicle routing, Capítulo 12 en [113] (1985) 431-448.

[45] Dantzig, G.B., Discrete variable extremum problems, *Operations Research* 5(2) (1957) 266-277.

[46] Dantzig, G.B. y P. Wolfe, The decomposition algorithm for linear programming, *Econometrica*, vol. 29 (1961) 767-768.

[47] Dantzig, G.B., *Linear Programming and Extensions*, Princeton University Press, Princeton-N.J., 1963.

[48] Dantzig, G., D. Fulkerson y S. Johnson, Solution of a large scale traveling salesman problem, Operations Research 2 (4) (1954) 393-410.

[49] Delgado, M., A resolution method for multiobjective problems, *Eur. J. of Oper. Research* 13 (1983) 165-172.

[50] Delgado, M., F. Herrera, J.L. Verdegay y M.A. Vila, Post-optimality analysis on the membership functions of a fuzzy linear programming problem, *Fuzzy Sets and Systems* 53 (1993) 289-297.

[51] Delgado, M., J.L. Verdegay y M.A. Vila, Fuzzy transportation problems: A general analisis, Kacprzyk and Orlovski (Eds*) Optimization Models Using Fuzzy Sets and Possibility Theory*, Reidel Dordrecht-Boston-Lancaster-Tokio (1987) 342-358.

[52] Delgado, M, J. L. Verdegay, M. A. Vila; A general model for fuzzy linear programming. *Fuzzy Sets and Systems* 29 (1989) 21-29.

[53] Delgado, M., J.L. Verdegay y M.A.Vila, Relating different approaches to solve linear programming problems with imprecise costs, *Fuzzy Sets and Systems* 37 (1990) 33-42.

[54] Dorfman, R., P.A. Samuelson y R.M. Solow, *Programación Lineal y Análisis Económico*, Segunda edición, Aguilar ediciones, Madrid, 1964.

[55] Dubois, D. Y H. Prade*; Fuzzy Sets and Systems: Theory and Applications*, Academic Press, New York, 1980.

[56] Dutta, D., R.N. Tiwari y J.R. Rao, Multiple objective linear fractional programming-A fuzzy set theoretic approach, *Fuzzy Sets and Systems* 52 (1992) 39-45.

[57] Dutta, D., J.R. Roa y R.N. Tiwari, Sensitivity analysis in fuzzy linear fractional programming problem, *Fuzzy Sets and Systems* 48 (1992) 211-216.

[58] Dutta, D., J.R. Rao y R.N. Tiwari, Fuzzy approaches for multiple criteria linear fractional optimization: A comment, *Fuzzy Sets and Systems* 54 (1993) 347-349.

[59] Edmonds, J., Matroids and greedy algorithm, *Mathematical Programming*, vol 1 (1971) pp. 127-136.

[60] Eiselt, H. A. Y G. Laporte, A combinatorial optimization problem arising in dartboard dcsign, *Journal of the Operational Research Society* 42 (1991) 113-118.

[61] Fabian, C., G. Ciobanu y M. Stoica, Interactive polyoptimization for fuzzy mathematical programming, in J. Kacprzyk y S.A. Orlovski (Eds.) *Optimization Models Using Fuzzy Sets and Possibility Theory* (D. Reidel, Dordrecht, 1987)272-291.

[62] Fabian, C. Y M. Stoica, Fuzzy integer programming, *In Fuzzy Sets and Decision Analysis*, H.J. Zimmermann, L.A. Zadeh y B.R. Gaines (Eds.), Amsterdam(1984) 123-132.

[63] Fedrizzi, M., J. Kacprzyk y M. Roubens (Eds), *Interactive Fuzzy Optimization, Lecture Notes in Economics and Mathematical Systems*, No. 368, Springer-Verlag. 1991.

[64] Fedrizzi, M. y R. Fullér, Stability in possibilistic linear programming with continuous fuzzy numbcr paramctcrs, *Fuzzy Sets and Systems* 47 (1992) 187-191.

[65] Feng Ying-Jun, A method using fuzzy mathematics to solve vectormaximum problcm, *Fuzzy Sets and Systems* 9 (1983) 129-136.

[66] Fullér, R., On stability in fuzzy linear programming problems, *Fuzzy Sets and Systems* 30 (1989) 39-344.

[67] García-Aguado, C. Y J.L. Verdegay, On the sensitivity of membership functions for fuzzy lincar programming problcms, *Fuzzy Sets and Systems* 56 (1993) 47-49.

[68] Garfinkel, R. S., Motivation and modelling, Cap.2 en [113] (1985) 17-36.

[69] Gass, S., *Programación Lineal: Métodos y aplicaciones*, ed. Continental, Mexico, 1985.

[70] Gay, D.M., A variant of Karmarkars linear programming algorithm for problems in standard form, *Mathematical programming* 37 (1987) 81-90.

[71] Gellinck(de), G.T. y J.P. Vial, A polinomial Newton method for linear programming, *Algorithmica* (1986) 1:125-153.

[72] Gilmore. P.C., E.L. Lawler y D.B. Shmoys, Well-solved special cases, Cap. 4 de [113] (1985) 87-143.

[73] Golden, B.L. y W.R. Stewart, Empirical analysis of heuristics, Cap. 7 en [113] (1985) 207-249.

[74] Goldfarb, D. Y S. Mehrotra, A relaxed version of Karmarkar's method, *Mathematical Programming* 40 (1988) 289-315.

[75] Goldfarb, D. Y S. Mehrotra, Relaxed variants of Karmarka's algorithm for linear programs with unknown optimal objective value, *Mathematical Programming* 40 (1988) 183-195.

[76] Gorry, G.M. y M.S. Scott-Morton, A Framework for management information systems, *Sloan Management Review*, Fall-1971.

[77] Hamacher, H., H. Leberling y H.J. Zimmermann, Sensitivity analisis in fuzzy linear programming , *Fuzzy Sets and Systems* 1 (1978) 269-281.

[78] Hannan, E. L., Linear programming with multiple fuzzy goals, *Fuzzy Sets and Systems*, 6 (1981) 235-248.

[79] Herrera, F., M. Kovács y J.L. Verdegay, Optimality for fuzzified mathematical programming problems: A parametric approach, *Fuzzy Sets and Systems* 54 (1993) 279-285.

[80] Herrera, F. y J.L. Verdegay, Aproaching fuzzy integer linear programming problems, In Fedrizzi, Kacprzyk and Roubens (Eds.), *Interactive Fuzzy Optimization* (Springer-Verlag, Berlin, 1991) 78-91.

[81] Herrera, F. Y J.L. Verdegay. Fuzzy control rules in optimization problems, *Scientia Iranica*, Vol 3, Nos.1,2,3. (1996) 89-96.

[82] Herrera, F., J.L. Verdegay y H.J. Zimmermann, Boolean programming problems with fuzzy constraints, *Fuzzy Sets and Systems* 55 (3)(1993) 285-293.

[83] Hoffman, A. J. y P. Wolfe, History, Cap. 1 en [113] (1985) 1-15.

[84] Horowtz, E., S. Sahni y S. Rajasekaran, *Computer Algorithms*, Computers Sc. Press,1998.

[85] Hulsurkar, S., M.P. Biswal y S.B. Sinha, Fuzzy programming approach to multi-objective stochastic linear programming problems, *Fuzzy Sets and Systems* 88 (1997) 173-181.

[86] Hussein, M.L. Qualitative analysis of basic notions in an iteractive approach to possibilistic goal programming, *Fuzzy Sets and Systems* 54 (1993) 39-46.

[87] Ida, K. y M. Gen, Improvemen of the two-phase approach to solving fuzzy multiple objective linear programing problems, *Japanese Journal of Fuzzy and Systems* vol 9 (1) (1997) 105-114.

[88] Ignizio, J.P., On the (re)discovery of fuzzy goal programming, *Decision Sci.* 13 (1982) 331-336.

[89] Ignizio, J.P., *Goal programming and extension* (Heath Lexington Bookc, London, 1976).

[90] Inuiguchi, M. y H. Ichihashi, Relative modalitoes and their use in possibilistic linear programming, *Fuzzy Sets and Systems* 35 (1990) 303-323.

[91] Inuiguchi, M., H. Ichihashi y Y. Kume, A solution algorithm for fuzzy linear programming with pieciwise linear memnership functions, *Fuzzy Sets and Systems* 34 (1990) 15-31.

[92] Inuiguchi, M., H. Ichihashi y Y. Kume, Relationships between modality constrained programming problems and various fuzzy mathematical programming problems, *Fuzzy Sets and Systems* 49 (1992) 243-259.

[93] Inuiguchi, M. y M. Sakawa, Possible and necesary optimality tests in possibilistic linear programming problems, *Fuzzy Sets and Systems* 67 (1994) 29-46.

[94] Inuiguchi, M. y M. Sakawa, Possible and necesary efficiency in possibilistic multiobjetive linear programming problems and possible eficiency test, *Fuzzy Sets and Systems* 78 (1996) 231-241.

[95] Iri, M. y H. Imai, A multiplicative barrier function method for linear programming, *Algorithmica* (1986) 1:455-482.

[96] Jimenez F. y J.L. Verdegay, Uncertain solid transportation problems, *Fuzzy Sets and Systems* 100 (1998) 45-57.

[97] Johnson, D.S. y C.H. Papadimitriou, Performance guarantees for heuristics, Cap. 5 cn [113] (1985) 145-180.

[98] Karloff, Howard, *Linear Programming*, Ed. Birkhäuser, Boston, USA, 1991.

[99] Karmarkar, N., A new polynomial-time algorithm for linear programming, *Combinatórica* 4 (1984), 373-395.

[100] Kato, K., M. Sakawa y T. Ikegame, Interactive decision making for multiobjective block angular 0-1 programming problems with fuzzy parameters trough genetic algorithms, *Japanese Journal of Fuzzy Theory and Systems* vol. 9 (1) (1997) 49-59.

[101] Keen, P.G.W. y M.S. Scott-Morton, *Decision Support Systems, An Organizational Perspective*, ponencia, MA:Addison-Wesley, 1978.

[102] Khachiyan, L. G., A polynomial algorithm for linear programming, *Soviet Mathematics Doklady* 20(1979), 191-194.

[103] Klee, V. y G. J. Minty, How good is the simplex Algorithm?, in Inequalities-III, O. Shisha, ed., Academic Press, New York, NY, 1972, 159-175.

[104] Kojima, M., Determining basic variables of optimal solutions in Karmarkar's new LP algorithm, *Algorithmica* (1986) 1:499-515.

[105] Lai, Y. J. Y Ch. L. Hwang, Interactive fuzzy linear programming, *Fuzzy Sets and Systems* 45 (1992) 169-183.

[106] Lai, Y.J. y Ch.L. Hwang, A new approach to some possibilistic linear programming problcms, *Fuzzy Sets and Systems* 49 (1992) 121-133.

[107] Lai, Y. J. Y Ch. L. Hwang, *Fuzzy Mathematical Programming, methods and aplications. Lecture notes in economics and mathematical systems* 394. Springer-Verlag, Berlin, 1992.

[108] Lai, Y. J. Y Ch. L. Hwang, IFLP-II: a decision support system, *Fuzzy Sets and Systems* 54 (1993) 47-56.

[109] Lai, Y.J. y Ch.L. Hwang, Possibilistic linear programming for managing interest ratc risk, *Fuzzy Sets and Systems* 54 (1993) 135-146.

[110] Lai, Y.J. y Ch.L. Hwang, A stochastic possibilistic programming model for bank hedging decision problems, *Fuzzy Sets and Systems* 57 (1993) 351-363.

[111] Laporte, G. The traveling salesman problem: An overview of exact and aproximate algoritms *European Journal of Operational Research*, 59 (1992) 231-247.

[112] Laporte, G. The vehicle routing problem: An overview of exact and aproximate algorithms, *European Journal of Operational Research*, 59 (1992) 345-358.

[113] Lawler E.L., J.K. Lenstra, A.H.G. Rinnooy Kan y D.B. Shmoys (editores), *The Traveling Salemsan Problem: A Guided tour of combinatorial optimization*, Jhon Wiley & Sons, Chichester, 1985.

[114] Leberling, H., On finding compromice solutions in multi-criteria problems using the fuzzy min-operator, *Fuzzy Sets and Systems* 6 (1981)105-118.

[115] Lee, E.S. y R.J. Li, Fuzzy multiple objective programming and compromise programming with pareto optimum, *Fuzzy Sets and Systems* 53 (1993)275-288.

[116] Lee, Y.R., Y. Shi y P.L. Yu, Linear optimal designs and optimal contingency plans, *Management Science* 36 (1990) 1106-1119.

[117] Lee, C-S y C-G. Wen, Fuzzy goal programming approach for water quality management in a river basin, *Fuzzy Sets and Systems* 89 (1997) 181-192.

[118] Lenstra, J.K. y A.H.G. Rinnooy Kan, Some simple applications of the traveling salesman problem, *Operational Research Quarterly* 26 (1975) 717-783.

[119] Li, H.L. y Ch. S. Yu, Comments on "fuzy programming with nonlinear membership functions...", Fuzzy Sets and Systems101 (1999) 109-113.

[120] Little, J.D.C., K.G. Murty, D.W. Sweeney y C. Karel, An algorithm for the traveling salesman problem, *Operations research* 11 (1963) 972-989.

[121] Liu, B. y K. Iwamura, Chance constrained programming with fuzzy parameters, *Fuzzy Sets and Systems* 94 (1998) 227-237.

[122] Liu, B. y K. Iwamura, A note on chance constrained programming with fuzzy coefficients, *Fuzzy Sets and Systems* 100 (1998) 229-233.

[123] Liu, Y.H. y Y. Shi, A fuzzy programming approach for solving a multiple criteria and multiple constraint level linear programming problem, *Fuzzy Sets and Systems* 65 (1994) 117-124.

[124] Luhandjula, M. K., Compensatory operators in fuzzy linear programming with multiple objectives, *Fuzzy Sets and Systems* 8(1982) 245-252.

[125] Luhandjula, M.K., Linear programming under randomness and fuzzines, *Fuzzy Sets and Systems* 10 (1983) 45-55.

[126] Luhandjula, M.K., Fuzzy approaches for multiple objective linear fractional optimizatión, *Fuzzy Sets and Systems* 13 (1984)11-23.

[127] Luhandjula, M.K., On possibilistic linear programming, *Fuzzy Sets and Systems* 18 (1986) 15-30.

[128] Luhandjula, M.K., Multiple objective programming problems with possibilistic coeficients, *Fuzzy Sets and Systems* 21 (1987)135-146.

[129] Lusting, I.J., Feasibility issues in a primmal-dual interior-point method for linear programming, *Mathematical Programming* 49 (1991) 145-162.

[130] Lusting, I.J., R.E. Marsten y D.F. Shano, Computational experience with a primal-dual interior point method for linear programming, *Linear Algebra and its Applications*, vol. 152 (1991) pp. 191-222.

[131] Malek, M., M. Guruswamy y M. Pandya, Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem, *Annals of Operations Research* 21 (1989) 59-84.

[132] Martello, S. y P. Toth, An upper bound for the zero-one knapsack problem and a branch and bound algorithm, *European Journal of Operational Research* 1 (1977) 169-175.

[133] Martello, S. y P. Toth, A new algorithm for the 0-1 knapsack problem, *Management Science* 34 (1988) 633-644.

[134] Martello, S. y P. Toth, *Knapsack Problems*, John Wiley and Sons 1990.

[135] Masaaki, I., Optimality on possibilistic linear programming with normal possibility distribution coefficient, *Japanese Journal of Fuzzy Theory and Systems* vol 7 (3) (1995) 349.

[136] Mitten, L.G. Branch-and-bound methods:general formulations and properties, *Operations Research* 18(1)(1970) 24-34.

[137] Mizunuma, H. y J. Watada, Fuzzy mixed integer programming based on genetic algorithm and its application to rcsourcc distribution, *Japanese Journal of Fuzzy Theory and Systems*, vol. 7 (1) (1995) 97-117.

[138] Mohamed, R.H., The relationship goal programming and fuzzy programming, *Fuzzy Sets and Systems* 89 (1997) 215-222.

[139] Monma, C.L. y A. J. Morton, Computational experiencewith a dual variant of Karmarkar's method for linear programming, *Operations Research Letters*, vol. 6, No. 6 (1987) pp. 261-267.

[140] Montciro, R.D.C. y I. Adlcr, Intcrior path following primal-dual algorithms. Part I: Linear programming, *Mathematical programming* 44 (1989) 27-41.

[141] Monteiro, R., I. Adler y M.G.C. Resende, A polinomial-time primal dual affine-scaling algorithm for linear and convex quadratic programming and its power series estensions, *Mathematics of Operations Research*, Vol. 15, No. 2 (1990) pp. 191-214.

[142] Nagata, M., T. Yamaguchi y Y. Kono, An interactive method for multi-period multiobjective production-transportation programming problems with fuzzy coeficients, *Japanese Journal of Fuzzy Theory and Systems* vol. 7 (1) (1995) 81.

[143] Nakahara, Y. y M. Gen, New relations of trapezoidal fuzzy numbers and its application to fuzzy linear programming problems, *Japanese Journal of Fuzzy Theory and Systems*, vol. 7 (6) (1995) 775-791.

[144] Narasimhan, R., Goal programming in a fuzzy enviroment, *Decision Sci.* 11 (1980) 325-336.

[145] Narasimhan, R., On fuzzy Goal programming-some comments, *Decision Sci.* 12 (1981) 532-538.

[146] Nazareth, J.L. Homotopy techniques in linear programming, *Algorithmica* (1986) 1: 529-535.

[147] Negoita, C. V. y M. Sularia, On fuzzy mathematical programming and tolerances in planning, *Economic Computer and Economic Cybernetic Studies and Researches* 1 (1976) 3-15.

[148] Óhéigeartaigh, M., A fuzzy transportation algorithm, *Fuzzy Sets and Systems* 8 (1982) 235-243.

[149] Ohta, H, y Yamaguchi T., Multi-goal programming including fractional goal in consideration of fuzzy solutions, *Japanese Journal of Fuzzy Theory and Systems* vol. 7 (6) (1995) 793.

[150] Okada, S. Y M. Gen, Fuzzy 0-1 knapsack problem, *Proceedings of the Fifth International Fuzzy System Association World Congress*, in Seoul, Korea (1993) 616-619.

[151] Okada, S. Y M. Gen, A method for solving fuzzy multidimensional 0-1 knapsack problcms, *Japanese Journal of Fuzzy Theory and Systems* vol. 6 (6) (1994) 687-702.

[152] Ollero, A., J. Aracil y E.F. Camacho, Optimization of dynamic regional models: an intcractivc multiobjcctivc approach, *Large Scale Systems* 6 (1984) 1-12.

[153] Orlovski, S.A., Multiobjective programming problems with fuzzy parameters, *In Kacprzyk* (1984), 175-184.

[154] Ostermark, R., Sensitivity analysis of fuzzy linear programs: an approach to parametric intcrdcpcndcncc, *Kibernetes* 16 (1987) 113-120.

[155] Papadimitriou, C.H. y K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, New Jersey, 1982.

[156] Prawda, J., *Métodos y Modelos de Investigación de Operaciones*, Tomo 1, Limusa, Mexico, 1995.

[157] Ramik, J. y H. Rommelfanger, A single and a multivalued order on fuzzy numbers and its use in linear programming with fuzzy coefficients, *Fuzzy Sets and Systems* 57 (1993) 203-208.

[158] Ravi, V. Y Reddy P.J. Fuzzy linear fractional goal programming applied to refinery operations planning, *Fuzzy Sets and Systems* 96 (1998) 173-182.

[159] Rinaldi, G. A projective method for linear programming with box-type constraints, *Algorithmica* (1986) 1: 517-527.

[160] Rodder, W. Y H.J. Zimmermann, Duality in fuzzy linear programming, In A.V. Fiacco and K.O. Kortanek (Eds), *Extremal Methods and Systems Analysis*, Springer-Verlag, Berlín, Heidelberg, New York (1980) 415-429.

[161] Rommelfanger, H., R. Hanuscheck y J. Wolf; Linear programming with fuzzy objectives. *Fuzzy Sets and Systems* 29 (1989), 31-48.

[162] Rosenkrantz, D.J., R.E. Stearns y P.M. Lewis II, An analysis of several heuristics for the traveling salesman problem. *SIAM J. Computing* 6 (1977) 563-581.

[163] Rubin, P.A. y R. Narasimhan, Fuzzy goal programming with nested priorities, *Fuzzy Sets and Systems* 14 (1984) 115-129.

[164] Sahni, S. Approximate algorithms for the 0-1 knapsack problem, *Journal of ACM* 22 (1975) 115-124.

[165] Sakawa, M. Fuzzy interactive multiobjetive programming, *Japanese Journal of Fuzzy Theory and Systems*, vol. 4 (1) (1992) 85.

[166] Sakawa, M., M. Iuiguchi y K. Sawada, Fuzzy satisfactionmethod using a convex fuzzy decision for large-scale linear programming problems with a block angular structure, *Japanese Journal of fuzzy Theory and Systems*, vol. 5 (3) (1993) 283-298.

[167] Sakawa, M., M. Inuiguchi y K. Sawada, A fuzzy satisficing method for large-scale multiobjective linear programming problems with block angular structure, *Fuzzy Sets and Systems* 78 (1996) 279-288.

[168] Sakawa, M y K. Kato, Interactive decision making for large-scale multiobjective linear programs with fuzzy numbers, *Fuzzy Sets and Systems* 88 (1997) 161-172.

[169] Sakawa, M. y K. Kato, Interactive decision-making for multiobjective linear fractional programming problems with block angular structure involving fuzzy numbers, *Fuzzy Sets and Systems* 97 (1998) 19-31.

[170] Sakawa, M., K. Kato y R. Mizouchi, Interactive decision-making for large scale multiobjective linear programming problems with fuzzy parameters, *Japanese Journal of Fuzzy Theory and Systems*, vol 7 (3) (1995) 377-390.

[171] Sakawa M. y K. Sawada, An interactive fuzzy satisficing method for large-scale multiobjective linear programming problems with block angular structure, *Fuzzy Sets and Systems* 67 (1994) 5-17.

[172] Sakawa, M. y H. Yano, An interactive fuzzy satisficing method for multiobjective linear fractional programming problems, *Fuzzy Sets and Systems* 28 (1988) 129-144.

[173] Sakawa, M. y H. Yano, An interactive fuzzy satisfacing method for generalized multiobjective linear programming problems with fuzzy parameters, *Fuzzy Sets and Systems* 35 (1990) 125-142.

[174] Sakawa, M. y H. Yano, Interactive decision making for multiobjective linear fractional programming problems with fuzzy parameters based on solution concepts incorporating fuzzy goals, *Japanese Journal of Fuzzy Theory and Systems* vol. 3 (1) (1991) 45.

[175] Salkin, H.M. y K. Mathur, *Foundations of integer programming*, North-Holland, New York, 1989.

[176] Sawada, K. y M. Sakawa, An interactive fuzzy satisficing method for large-scale multiobjective linear programming problems with block angular structure, *Japanese Journal of Theory and Systems* vol. 6 (4) (1994) 373-387.

[177] Seiford , L. y P. L. Yu, Potential solutions of linear systems; the multi-criteria multiple contraint level program, *Journal of Mathematical Analysis and Applications* 69 (1979) 283-303.

[178] Seo, F. y M. Sakawa, *Multiple Criteria Decision Analysis in Regional Planning:Concepts, Models and Applications* (D. Reidel, Dordrecht, 1988)

[179] Shi, Y. y Y.H. Liu, Fuzzy potential solutions in multicriteria and multiconstraint level linear programming problems, *Fuzzy Sets and Systems* 60 (1993)163-179.

[180] Simon, H., *The New Science of Management Decision*, Englewood Cliffs, NJ, Prentice-Hall, 1977.

[181] Slowinski, R., A multicriteria fuzzy linear programming method for water supply system development planning, *Fuzzy Sets and Systems* 19 (1986) 217-237.

[182] Slowinski, R., An interactive method for multiobjective linear programming with fuzzy parameters and its applications to water supply planning, in J. Kacprzyk y S.A. Orlovski (Eds), *Optimization Models using Fuzzy Sets and Possibility theory* (D. Reidel, Dordrecht, 1987) 396-414.

[183] Slowinski, R. y J. Teghem (Eds), *Stochastic versus Fuzzy Approaches to Multiobjective Mathematical Programming under Uncertainty*, D. Reidel Publishing Co. 1990.

[184] Smith, J.D. *Design and Analysis of Algorithms*, Pws-Kent p. co, Boston, 1989.

[185] Stancu-Minasian, I.M., *Stochastic Programming with Multiple Objective Functions* (D. Reidel, Dordrecht, 1984).

[186] Sularia, M., On fuzzy programming in planning, *Kybernets* 6 (1977) 229-230.

[187] Tanaka, H. Y K. Asai; Fuzzy linear programming problems with fuzzy numbers. *Fuzzy Sets and Systems* 13 (1984) 1-10.

[188] Tanaka, H., H. Ichihashi y K. Asai; A formulation of fuzzy linear programing problems basad on comparison of fuzzy numbers. *In Kacprzyk* (1984), 185-194.

[189] Tanaka, H., H. Ichihashi y K. Asai, A value of information in FLP problems via sensitivity analysis, *Fuzzy Sets and Systems* 18 (1986) 119-129.

[190] Tanaka, H., T. Okuda y K. Asai, On fuzzy mathematical programming, *Journal of Cybernetics* 3 (1974) 37-46.

[191] Teng, J-Y. y G-H. Tzeng, Transportation investment project selection using fuzzy multiobjective programming, *Fuzzy Sets and Systems* 96 (1998) 259-280.

[192] Tiwari, R.N., S. Dharmar y J.R. Rao, Priority structure in fuzzy goal programming, *Fuzzy Sets and Systems* 19 (1986)251-259.

[193] Tiwari, R.N., S. Dharmar y J.R. Rao, Fuzzy goal programming - an additive model, *Fuzzy Sets and Systems* 24 (1987)27-34.

[194] Todd, M.J. y B.P. Burrell, An extension of Karmarkar's algorithm for linear programing using dual variables, *Algorithmica* (1986) 1: 409-424.

[195] Todd, M.J. y Y. Ye, A centered proyective algorithm for linear programming, *Mathematics of Operations Research*, Vol 15 (1990) pp. 508-529.

[196] Tong S., Interval number and fuzzy number linear programmings, *Fuzzy Sets and Systems* 66 (1990) 301-306.

[197] Turban. E., *Decision Suport and Expert Systems: Management Support Systems*, Fourth edition, Englewood Cliffs-NJ , Prentice Hall, 1995.

[198] Valenzuela, Ch.L. y A.J. Jones, Evolutionary divide and conquer (I): A novel genetic approach to the TSP, *Evolutionaruy Computation* 1 (4) (1994) 313-333.

[199] Vanderbei, R.J., Affine-scaling for linear programs with free variables, *Mathematical Programming* 43 (1989) 3-44.

[200] Vanderbei, R.J., M.S. Meketon y B.A. Freedman, A modification of Karmarkar's linear programming alghorithm, *Algorithmica* (1986) 1:395-407.

[201] Verdegay, J.L. Problemas de transporte con parámetro difuso, *Rev. Acad. Ciencias Mat. Fis. Quim. Y Nat.* De Garanada, 2 (1983) 47-56.

[202] Verdegay, J. L., A dual approach to solve the fuzzy linear programming problem, *Fuzzy Sets and Systems* 14(1984) 131-141.

[203] Verdegay, J.L. Fuzzy mathematical programming, *Approximate Reasoning in Decision Analysis* - Gupta, M.M. y E. Sanchez (eds) (North_holland, A,sterdam, 1992).

[204] Verdegay, J. L.; Fuzzy optimization: Models, methods and Perspectives; *6thIFSA-95 World Congress.* Sou Paulo - Brazil, 1995, pp. 39-71.

[205] Verma, R., M.P. Biswal y A. Biswas , Fuzzy programming technique to solve multi-objective transportation problems with some non-linear membership functions, *Fuzzy Sets and Systems* 91 (1997) 37-43.

[206] Wang, H.F. y M.L. Wang, A fuzzy multiobjective linear programming, *Fuzzy Sets and Systems* 86 (1997) 61-72.

[207] Wang, G-Y. y Q. Zhong, Linear programming with fuzzy random variable coeficients, *Fuzzy Sets and Systems* 57 (1993) 295-311.

[208] Werners, B., An interactive fuzzy programming system, *Fuzzy Sets and Systems* 23 (1987) 131-147.

[209] Werners, B., Interactive multiobjective programming subject to flexible constraints, *European J. Oper. Res.* 31 (1987) 342-349.

[210] Wets, R.J.B. , Stochastic Programming, in G.L. Nemhauser, A. H.G. Rinnooy Kan y M.J. Todd (Eds.), *Handbooks in Operations Research and Management Science: Optimization* (North-holland, Amsterdam, 1989) 573-629.

[211] Yamaguchi, T. e Y. Kono, Application of fuzzy multiobjective linear programming to greenhouse cultivation planning, *Japanese Journal of Fuzzy Theory and Systems* vol 4 (6) (1992) 701.

[212] Yang, T. Y J.P. Ignizio, Fuzzy programming with nonlinear membership functions: piecewise linear approximation, *Fuzzy Sets and Systems* 41 (1991)39-53.

[213] Yano, H. Y M. Sakawa, Interactive fuzzy decision making for generalized multiobjective linear fractional programming problems with fuzzy parameters, *Fuzzy Sets and Systems* 32 (1989) 245-261.

[214] Ye, Y. Y M. Kojima, Recovering optimal dual solutions in Karmarkar's polynomial algorithm for linear programming, *Mathematical Programming* 39 (1987) 305-317.

[215] Yokoyama, T., H. Ohta y T. Yamaguchi, Multiobjective probabilistic constrained programming problems using fuzzy goal, *Japanese Journal of Fuzzy Theory and Systems* vol 6 (6) (1994)

[216] Zadeh, L.A.: Fuzy sets, *Information and Control* 8 (1965) 338-353.

[217] Zadeh, L.A., Fuzzy sets as a basis para una theory of possibility, *Fuzzy Sets and Systems* 1 (1978) 3-28.

[218] Zeleny, M., Compromise programming, in: J.L. Cochrane and M. Zeleny, Eds. *Multiple Criteria Decision Making*, University of South carolina Prcs (1973).

[219] Zhao, R., R. Govind y G. Fan, The complete decision set of the generalized symetyrical fuzzy linear programming problem, *Fuzzy Sets and Systems* 51 (1992) 53-65.

[220] Zhong, Q. Y G-Y. Wang, On solutions and distribution problems of the linear programming with fuzzy random variable coeficients, *Fuzzy Sets and Systems* 58 (1993) 155-170.

[221] Zhong, Q., Y. Zhang y G-Y. Wang, On fuzzy random linear programming, *Fuzzy Sets and Systems* 65 (1994) 31-49.

[222] Zimmermann, H. J., Description and optimization of fuzzy system, *International Journal of general System* 2 (1976) 209-216.

[223] Zimmermann, H. J., Fuzzy programming and linear programming with several objective functions, *Fuzzy Sets and Systems* 1 (1978) 45-55.

[224] Zimmermann H.J., *Fuzzy Sets, Decision Making y Sistemas Expertos* (Kluwer Academic, Boston , 1987).

[225] Zimmermann H.J. y M.A. Pollatschek, Fuzzy 0-1 linear programs, *In H.J. Zimmermann, L.A. Zadeh and B.R.Gaines* (Eds.), Amsterdam (1984) 133-145.

[226] Zionts, S. Y J. Wallenias, An interactive programming method for solving the multiple criteria problems, *Management Sci.* 22 (1976) 652-663.