LA TECNICA DE LA BUSQUEDA TABU Y SUS APLICACIONES

SERGIO GERARDO DE LOS COBOS SILVA

TESIS DOCTORAL

PRESENTADA A LA DIVISION DE ESTUDIOS DE
POSGRADO DE LA
FACULTAD DE INGENIERIA
DE LA
UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

COMO REQUISITO PARA OBTENER
EL GRADO DE
DOCTOR EN INGENIERIA

DIRECTOR DE TESIS
DR. MIGUEL ANGEL GUTIERREZ ANDRADE

CIUDAD UNIVERSITARIA
1994

A mis Padres Celestiales, a quienes todo les debo.

A mi querida Madre, por su amor, apoyo y comprensión desinteresada.

A mi Amada Esposa Arcelia y a mis Hijos: Sergio, Arcelia, Montserrat y Josemaría, por todo su amor y por todas las horas robadas.

A México, mi querida Patria.

Deseo agradecer de manera muy especial a los Señores Doctores: Sergio Fuentes Maya, Francisco Venegas Martínez y Miguel Angel Gutiérrez Andrade, que sin sus valiosos consejos, excelente y profesional dirección en mis estudios y no pocas veces su apoyo y estímulo personal, no hubiera alcanzado esta meta.

También deseo agradecer a todos los demás miembros del Jurado Doctoral y a todos los profesores de la DEPFI que me proporcionaron sus conocimientos.

Agradezco al Dr. Fred Glover de la Universidad de Colorado por la información e interés proporcionado.

Deseo también agradecer la amistad de todos mis compañeros, a los cuales los considero como amigos por siempre.

El presente trabajo se realizó mediante el apoyo de(en orden alfabético):

Consejo Nacional de Ciencia y Tecnología, (CONACyT), México.

División de Estudios de Posgrado, Facultad de Ingeniería, UNAM. (Jefatura de la División, Secretaría de la División y Jefatura del Departamento de Sistemas.)

Programa de Apoyo a las Divisiones de Posgrado, PADEP-UNAM.

Universidad Autónoma Metropolitana Unidad Iztapalapa, (Consejo Divisional de Ciencias Básicas e Ingeniería, Departamento de Matemáticas, Programa de Formación de Personal Académico.

RESUMEN

En las últimas décadas se ha presentado una variedad importante de problemas de optimización combinatoria de gran escala. Las aplicaciones van desde la localización de centros de emergencia hasta los más sofisticados diseños de modelos industriales, económicos y militares, dichos problemas presentan gran complejidad en la determinación de soluciones óptimas o cercanos a la solución óptima.

Hoy en día, el reto para varias disciplinas es proponer técnicas eficientes de solución, que por un lado garanticen soluciones "buenas", que sean rápidos y fáciles de implantar.

Una forma de tratar con este tipo de problemas, es ir un paso adelante de la destreza y del conocimiento del experto. Muchos de los algoritmos existentes, presentan la dificultad de escapar de la optimalidad local y/o del ciclado. Nosotros presentamos la técnica conocida como *búsqueda tabú*, cuya filosofía es la de manejar y explotar de manera eficiente una colección de principios en forma inteligente que en la actualidad presenta resultados de los más promisorios en la literatura para encontrar soluciones eficientes para los problemas combinatorios de gran escala. Esta técnica tiene la característica, entre otras, de salir de entrampamientos suboptimales y de realizar búsquedas en diferentes regiones del espacio de soluciones.

Nosotros aplicamos la búsqueda tabú para resolver el problema de calendarización de trabajos con costos de penalización y de actualización. A través de ejemplos, se presentán y describen diferentes elementos de la búsqueda tabú.

Proporcionamos direcciones de mejoramiento sobre el esfuerzo computacional para el problema de asignación cuadrático, mediante el desarrollo de diferentes algoritmos originales de la búsqueda tabú. Se incluyen comparaciones y contrastes entre ellos.

Nostros proponemos un método específico de la búsqueda tabú en la versión de demanda determinística con costo de reaprovisionamiento conjunto para el problema de inventarios multiproducto. Se describe el impacto de nuestro método con respecto de algoritmos especializados descritos como altamente eficientes en la literatura.

En suma, el propósito de este trabajo, es el de integrar algunas de las formas de caracterizar la búsqueda tabú y su aplicación en la solución de algunos problemas importantes en optimización combinatoria con ejemplos. Así también, señalamos una gran

variedad de direcciones para nuevas aplicaciones e investigaciones, nuestro desarrollo incluye comparaciones entre varios algoritmos conocidos para atacar diferentes problemas combinatorios de interés tanto teórico como práctico.

CONTENIDO

INTRODUCCION

En optimización combinatoria, uno de los problemas computacionales a resolver es la *explosión combinatoria.* La explosión combinatoria se encuentra en situaciones donde las elecciones están compuestas secuencialmente, es decir, dado un conjunto de elementos se pueden obtener diferentes arreglos ordenados de estos, permitiendo una vasta cantidad de posibilidades, por ejemplo, en la elección de rutas alternativas en un laberinto. Situaciones de este tipo ocurren en problemas de inversión financiera, operación de manufactura, manejo de inventarios, localización de recursos, presupuestación de capitales, etc. Algunos otros ejemplos incluyen la explotación de minerales, análisis de datos metereológicos, diseño de circuitos integrados, manejo de recursos hidráulicos, operaciones de monitoreo de satélites, y mantenimiento de sistemas.

Vale la pena mencionar el trabajo de Gutiérrez Andrade (1991), en el que se contrasta lo simple y complejo de este tipo de problemas:

> "Una característica recurrente en los problemas de optimización combinatoria es el hecho de que son muy "fáciles" de entender y de enunciar, pero generalmente son "difíciles" de resolver. Podría pensarse que la solución de un problema de optimización combinatoria se restringe únicamente a buscar de manera exhaustiva el valor máximo o mínimo en un conjunto finito de posibilidades y que usando una computadora veloz, el problema carecería de interés matemático, sin pensar por un momento, en el tamaño de este conjunto."

Los intentos por tratar con el problema de la explosión combinatoria han encontrado muchos obstáculos. Por ejemplo, no es suficiente contar con un "conocimiento experto" para manejarlos de manera efectiva, de igual manera no es suficiente confiar en el poder computacional de alta velocidad de las super computadoras. Algunos problemas clásicos donde la explosión combinatoria prevalece, muestran que un intento por generar todas las alternativas relevantes por computadora no es una tarea factible. Así, por ejemplo, en el problema del agente viajero, en el cual se tiene que salir de una ciudad y regresar a la misma después de haber visitado (con costo mínimo de viaje) todas las demás ciudades, si se tienen n ciudades en total que recorrer entonces existen $n!$ soluciones factibles, y si una computadora que pudiera ser programada para examinar soluciones a razón de un billón de soluciones por segundo; la computadora terminaría su tarea, para $n =$

24 ciudades (que es un problema pequeño para muchos casos prácticos) en alrededor de 19,674 años.

Al respecto Glover (1989) expone:

> "la clave para tratar con tales problemas es ir un paso más alla de la aplicación directa de la destreza y del conocimiento del experto, y generar recursos para un procedimiento especial (o marco) el cual monitorea y dirige el uso de esta habilidad y conocimiento. Careciendo de tal procedimiento, las reglas expertas se pueden *empantanar*, permitiendo un punto donde ninguna mejora puede percibirse, a menos de que existan alternativas superiores."

Aunque para algunos problemas de optimización como son los problemas de programación lineal, los problemas de transporte, los problemas de asignación, existen algoritmos rápidos y eficientes para muchos otros problemas de optimización combinatoria, no los hay como son los casos del problema de asignación cuadrático, el problema de localización de plantas, el problema del agente viajero, etc. Así, desde hace varios años estos últimos problemas han sido atacados por algoritmos desarrollados especialmente para el problema específico y usando una diversidad de técnicas tales como planos de corte, ramificación y acotamiento, enumeración implícita, relajación Lagrangeana, partición de Benders, etc. o por combinaciones de las técnicas antes mencionadas. Sin embargo, no pueden resolverse de manera exacta usando tiempo y espacio de computadora, aún cuando se tenga sólo un número moderado de variables. En la actualidad, la investigación se ha dirigido hacia el diseño de buenas heurísticas, *i.e.*, algoritmos eficientes con respecto al tiempo de cómputo y al espacio de memoria, y con cierta verosimilitud de entregar una solución "buena" *i.e.*, relativamente cercana a la óptima mediante el examen de solo un pequeño subconjunto del número total de permutaciones posibles.

El principal problema de algunos algoritmos heurísticos es la dificultad de escapar de la optimalidad local. Lo anterior ha propiciado que el enfoque de la inteligencia artificial haya revivido como solución de problemas que requieren de la búsqueda heurística. Recientemente varias aproximaciones han surgido del manejo de problemas de decisión complejos, como son: *algoritmos genéticos, redes neuronales, recocido simulado, búsqueda tabú, análisis de objetivos y búsqueda dispersa*, entre otros.

El propósito de este trabajo es presentar un algoritmo altamente eficiente en la solución de problemas de optimización combinatoria denominado *búsqueda tabú*. En

este se presenta primero, el estado del arte alrededor de los métodos más eficientes para resolver problemas de tipo combinatorio, y después se proporcionan aplicaciones concretas que muestran la bondad e importancia de la técnica. Las aportaciones realizadas en este trabajo son las siguientes:

- Revisión del estado del arte sobre las técnicas más eficientes utilizadas y que hasta ahora reportan los mejores resultados para atacar los problemas de programación matemática y en particular problemas de programación combinatoria.

- Desarrollo de un algoritmo de búsqueda tabú para el problema de calendarización con costos de penalización, aquí se muestran los diversos elementos de la técnica de búsqueda tabú, como son, la intensificación y diversificación regional de búsqueda.

- Desarrollo de tres algoritmos de la técnica de la búsqueda tabú, los cuales son originales en sus características para el problema de asignación cuadrático.

- Se reporta la experiencia computacional sobre la eficiencia de estos algoritmos.

- Se propone un nuevo algoritmo para el problema de inventarios multiproducto y se compara contra algoritmos considerados altamente eficientes en la literatura especializada, se reporta la experiencia computacional donde se muestra mayor eficiencia de nuestro algoritmo propuesto en todos los casos.

- Difundir la técnica de búsqueda tabú, mediante la integración de algunas formas fundamentales que la caracterizan.

- Señalar varias direcciones posibles de investigación y aplicación.

La Búsqueda Tabú es un procedimiento heurístico cuyas ideas básicas fueron esbozadas inicialmente por Fred Glover (1986) y por Pierre Hansen (1986) e introducido en su forma actual por Fred Glover (1989, 1990a), el cual es utilizado para resolver problemas de optimización de gran escala, dicho procedimiento heurístico está diseñado

para guiar a otros métodos (o a procesos componentes) para escapar de la optimalidad local. La búsqueda tabú ha obtenido soluciones optimales o cercanas de la óptima en una amplia variedad de problemas clásicos.

La búsqueda tabú, junto con la técnica del recocido simulado y los algoritmos genéticos, ha sido singularmente calificada por el Committee on Next Decade of Operations Research (Condor (1988)) como "extremadamente promisorio" para el tratamiento futuro de aplicaciones prácticas.

La búsqueda tabú tiene como antecedentes a varios métodos diseñados para cruzar fronteras de factibilidad o de optimalidad local; sistemáticamente impone y relaja restricciones para permitir la exploràción de regiones de otra manera prohibidas. Ejemplos iniciales de tales procedimientos incluyen heurísticas basadas sobre métodos de restricciones subrogadas[1] y aproximaciones de planos de corte que violan de manera sistemática las condiciones de factibilidad.

En general se puede decir que la filosofía de la búsqueda tabú está basada en manejar y explotar una colección de principios para resolver problemas de manera inteligente, la cual tiene como elemento principal el uso de memoria flexible. Desde cierto punto de vista, el uso de memoria flexible involucra el proceso dual de crear y explotar estructuras para tomar ventaja de la "historia", por lo que, se combinan las actividades de adquisición y mejoramiento de la información.

La búsqueda tabú se funda en tres puntos principales:

(1). El uso de estructuras de memoria basadas en atributos diseñados para permitir criterios de evaluación e información de búsqueda histórica, la cual se explota más a fondo que las estructuras de memoria rígida (como en ramificación y acotamiento) o por sistemas de pérdida de memoria (como recocido simulado y otros métodos aleatorizados).

(2). Un mecanismo asociado de control, mediante el empleo de estructuras de memoria, basado en el interjuego entre las condiciones que restringen y liberan al proceso de búsqueda (envuelto en las restricciones tabú y el criterio de aspiración).

(3). La incorporación de funciones de memoria de diferentes lapsos de tiempo, desde término corto hasta de término largo, para implantar estratégias que reforcen la combi-

[1]Subrogar: Sustituir a otro en un derecho o en una obligación.

nación de movimientos y las características de solución que históricamente se han encontrado buenas, mientras que las estratégias de diversificación manejan la búsqueda dentro de nuevas regiones.

Las estructuras de memoria de la búsqueda tabú operan bajo cuatro dimensiones principales: pertenencia, frecuencia, calidad e influencia. El papel de estos elementos en la creación de procesos efectivos para resolver problemas son uno de los focos de atención de este trabajo. En este trabajo se desarrollan los elementos descritos mediante la elaboración de diferentes algoritmos originales altamente eficientes para atacar problemas clásicos considerados en la literatura como difíciles.

Este trabajo se desarrolla de' la siguiente manera: En el capítulo 1, se describen los elementos principales de un sistema de inteligencia artificial de los problemas de representación y de las heurísticas; también se tratan diferentes técnicas que actualmente se utilizan para atacar eficientemente a problemas de optimización combinatoria como son el análisis de objetivos, los algoritmos genéticos y la búsqueda dispersa. En el capítulo 2, se formula el problema general de optimización combinatoria; se introduce la técnica de la búsqueda tabú, se establecen algunos conceptos y definiciones y se describen los elementos principales de ésta mediante el desarrollo de algunos ejemplos de calendarización. Se discute la necesidad de la intensificación y diversificación regional de la búsqueda. En el capítulo 3, se introducen varios modelos de problemas de localización, se describe y analiza el problema de asignación cuadrático así como la implantación de tres esquemas originales del método de búsqueda tabú para su solución y se presenta un análisis comparativo presentando la experiencia computacional. En el capítulo 4, se describe el modelo de inventarios multiproducto así como algũnos algoritmos que resuelven el problema con eficiencia, se propone un algoritmo alternativo el cual utiliza a la búsqueda tabú como metaprocedimiento de mejora y se presenta la experiencia computacional. En este mismo capítulo se describen algunos métodos para el manejo de la lista tabú de tipo dinámico. Finalmente, en las conclusiones se presentan los alcances, limitaciones y resultados más relevantes, así como posibles líneas de investigación.

CAPITULO 1

INTELIGENCIA ARTIFICIAL

Para la solución de problemas complejos combinatorios que tradicionalmente son de gran interés para la investigación de operaciones, han surgido diferentes enfoques de solución. Uno de los grandes desafios actuales de la inteligencia artificial es la explosión combinatoria. En este capítulo se realiza una revisión del estado del arte sobre los aspectos y enfoques más relevantes de ambas disciplinas, así como de las técnicas más eficientes que hasta ahora reportan los mejores resultados para atacar problemas de programación matemática.

El marco de nuestro trabajo involucra el uso de reglas condicionales para guiar la búsqueda de soluciones. Ahora bien, desde una perspectiva amplia se puede concebir a cualquier sistema como "experto" siempre que se diseñe con base en los procesos de aprendizaje de la Inteligencia Artificial y que adaptativamente exploten la memoria.

En el dominio de las heurísticas para la optimización combinatoria, los problemas en sí envuelven elecciones de ramificación múltiple codificadas por el uso de variables de valores discretos, o de forma que requieren la realización de decisiones secuenciales y comparaciones de alternativas.

El capítulo se desarrolla como sigue: en la sección 1.1, se presentan los elementos principales de un sistema de inteligencia artificial, así como una breve revisión sobre los problemas de representación; en la sección 1.2, se describe el análisis de objetivos, el cual se considera como un procedimiento que integra los marcos de conocimiento de la investigación de operaciones y de la inteligencia artificial; en la sección 1.3, se discuten los elementos principales de los algoritmos genéticos. Finalmente, en la sección 1.4, se proporciona una introducción a la relajación matemática y al método de búsqueda dispersa.

1.1 SISTEMAS DE INTELIGENCIA ARTIFICIAL

En Nilson (1987) se observa que los elementos principales en un *sistema de inteligencia artificial*, son una *base de datos global*, un conjunto de *reglas de inteligencia artificial* y

un *sistema de control.*

La base de datos global es la estructura central de datos usada en un sistema de inteligencia artificial.

Las reglas de inteligencia artificial operan sobre la base de datos global. Cada regla tiene una *precondición* que puede ser satisfecha o no por la base de datos global. Si la precondición es satisfecha, la regla puede ser aplicada. La aplicación de una regla produce un cambio en la base de datos. El sistema de control escoge qué reglas aplicables deben ser aplicadas y suspende el proceso cuando la base de datos global satisface una *condición de terminación.*

Considérese el juego popular llamado rompecabezas-8 el cual consiste en ocho fichas numeradas que pueden moverse sobre un tablero de 3 x 3 cuadros (ver fig. 1.1). Uno de los cuadros del tablero queda siempre vacío, lo que hace posible mover sobre él una de las fichas numeradas adyacentes, lo que podría interpretarse también como un movimiento del cuadro vacío. La figura 1.1 muestra dos configuraciones de fichas. Considérese el problema de cambiar una configuración inicial en una configuración determinada llamada configuración objetivo. Una solución al problema consiste de una secuencia adecuada para conseguirlo.

3	4	8		4	2	3
1	6	2	$\Longrightarrow$	8		7
7		5		1	6	5

Figura 1.1 Configuración inicial y objetivo para el rompecabezas-8.

Para resolver el problema se considerará un sistema de inteligencia artificial, por lo que se debe de especificar *los estados, los movimientos y el objetivo* del problema. En este caso, cada configuración es un estado del problema; el conjunto de todas las configuraciones posibles es el espacio de estados del problema el cual para nuestro ejemplo tiene sólo 362,880 (9!) configuraciones posibles.

Un movimiento transforma un estado del problema en otro. El rompecabezas-8 se interpreta convenientemente con base en los siguientes cuatro movimientos: mover el cuadro vacío (blanco) a la izquierda, a la derecha, hacia arriba y hacia abajo. Estos movimientos son modelados por reglas de inteligencia artificial que operan en forma

apropiada sobre las descripciones de los estados. Cada una de las reglas tiene condiciones previas que deben ser satisfechas por la descripción de un estado para que le sea aplicable; así, la condición previa para la regla asociada con "mover el blanco hacia arriba" es consecuencia de la exigencia de que ese espacio blanco no esté ya en la fila superior del tablero.

La condición objetivo del problema sirve de base para la condición de terminación del sistema de inteligencia artificial. La estrategia de control aplica repetidamente reglas a las descripciones de los estados hasta que se produce la descripción de un estado objetivo.

En ciertos problemas, se desea además que la solución esté sujeta a ciertas restricciones adicionales. Por ejemplo, se puede desear que la solución tenga un número mínimo de movimientos. En general se le atribuye un *costo* a cada movimiento y se intenta obtener una solución que tenga costo total mínimo.

Procedimiento Básico.

DATOS ⟵ Base de datos inicial

hasta que DATOS satisface la condición de terminación, **hacer:**

empezar

seleccionar alguna regla R, que pueda ser aplicada a DATOS.

DATOS ⟵ resultado de aplicar R a DATOS.

fin.

El procedimiento anterior no es determinístico, porque todavía no se ha especificado con precisión cómo se va a seleccionar una regla aplicable.

Las normas para seleccionar y para anotar aquellas secuencias de reglas ya intentadas y las bases de datos que produjeron, constituyen lo que se conoce como la estrategia de control para el sistema de inteligencia artificial. En muchas aplicaciones de inteligencia artificial (IA), la información disponible por la estrategia de control no es suficiente para determinar la regla más apropiada en cada paso. La operación de un sistema de

inteligencia artificial puede caracterizarse así, como un proceso de búsqueda, en el cual, las reglas se van probando hasta que se encuentra que alguna secuencia de ellas produce una regla de datos que satisface la condición de terminación. Las estrategias de control eficientes requieren un conocimiento del problema que ha de resolverse, suficiente para que la regla seleccionada en la ejecución de la selección tenga una gran probabilidad de ser la más apropiada.

Se pueden distinguir dos clases principales de estrategias de control: *irrevocables y tentativas.* En un régimen de control irrevocable, una regla aplicable que se selecciona es aplicada irrevocablemente, sin que sea posible una reconsideración posterior. En un régimen de control tentativo se selecciona una regla aplicable y se aplica, pero considerando medidas precisas para poder retornar a ese punto del proceso y entonces aplicar otra regla.

También se pueden distinguir dos tipos de regímenes de control tentativos. El primero, que llamaremos *retroactivo*, en cuyo caso, cuando se selecciona una regla, se establece un *punto de regreso.* Si en el proceso subsiguiente se encuentra alguna dificultad para llegar a una solución, el estado del proceso se retrae a ese punto de regreso y el proceso continúa aplicando otra regla distinta.

En el segundo tipo de control tentativo, que llamaremos de *exploración de grafos*, se adoptan medidas para registrar los efectos de varias secuencias de reglas simultáneamente. En este tipo de control se usan varias clases de estructuras de grafos y de exploración de éstos.

1.1.1 PROBLEMAS DE REPRESENTACION

Para la solución de un problema no basta con una estrategia de control eficiente; hace falta además seleccionar buenas representaciones de los estados del problema, movimientos y condiciones de terminación. La representación de un problema tiene una influencia muy grande en el esfuerzo que se invierte en resolverlo. Evidentemente serán preferibles representaciones con espacios de estados lo más reducidos posibles. Por ejemplo, hay muchas rompecabezas aparentemente difíciles que tienen espacios de estados trivialmente pequeños si se representan en forma adecuada. A veces un espacio de estados dado puede reducirse sustancialmente al descubrir que se pueden descartar ciertas reglas o combinar algunas de ellas para obtener otras más poderosas. Incluso, cuando no es posible hacer transformaciones de este tipo tan simple, una formulación del problema

totalmente nueva (cambiando, por ejemplo, la propia noción de lo que constituye un estado) puede conducir a un espacio de estados más reducido.

El proceso que se requiere para formular la representación inicial de un problema o para mejorar una representación dada, aún no está bien comprendida. Parece que el hallazgo de cambios deseables en la representación de un problema, depende de la experiencia acumulada en los intentos de resolverlo en una representación dada. Esta experiencia nos permite descubrir la existencia de ciertas circunstancias simplificadoras, tales como simetrías, o de secuencias de reglas útiles, que pueden combinarse formando *macro reglas.*

Como un ejemplo de los conceptos anteriores considérese el problema del agente viajero que tiene que visitar las ciudades A, B, C, D y E, además se supone que existe una carretera entre cualesquiera dos de estas ciudades con un costo asociado. El problema como ya se ha mencionado consiste en encontrar una ruta de costo mínimo, que partiendo de la ciudad A, visite cada una de las ciudades una sola vez y regrese a A.

Para plantear este problema se especificará lo siguiente:

> La base de datos global será la lista de ciudades visitadas hasta el momento. Así, la base de datos inicial se describe con la lista (A). No se permiten listas que contengan el nombre de una ciudad más de una vez, con excepción de que A puede aparecer otra vez al final de la lista cuando ella ya contiene antes todas las ciudades.
>
> Las reglas corresponden a las decisiones (*a*) ir a continuación a la ciudad A, (*b*) ir a continuación a la ciudad B, ... y (*e*) ir a continuación a la ciudad E. Una regla no es aplicable a la base de datos si no la transforma en otra legal. Así la regla "ir a continuación a la ciudad A" no será aplicable a una lista que no contenga ya los nombres de todas las ciudades.
>
> Sólo las bases de datos que empiecen y terminen con A e incluyan los nombres de todas las ciudades restantes pueden satisfacer la condición de terminación. Un viaje propuesto como solución debe ser de mínima longitud.

Los sistemas eficientes de IA requieren conocimiento del dominio del problema. Podemos subdividir, este conocimiento en tres amplias categorías que corresponden a las bases

de datos globales, a las reglas y al control de los sistemas de inteligencia artificial. El conocimiento acerca de un problema que se representa en la base de datos global se denomina a veces *conocimiento declarativo*. Por ejemplo, en un sistema inteligente de recuperación de información, el conocimiento declarativo incluirá a la base de datos principal de los hechos específicos. El conocimiento que se representa en forma de reglas suele llamarse procesal. En un sistema inteligente de recuperación de información, el conocimiento procesal incluirá los conocimientos generales que nos permitan manipular el conocimiento declarativo. El conocimiento que se representa mediante la estrategia de control se denomina frecuentemente *conocimiento de control*; éste incluye conocimientos acerca de diversos procesos, estrategias y estructuras que se usan para coordinar el proceso completo de resolución del problema.

1.1.2 HEURISTICAS

Las *Heurísticas* son criterios, métodos o principios para decidir sobre varias alternativas de cursos de acción que prometen ser lo más efectivo en orden a alcanzar alguna meta. Ellas representan compromisos con dos requerimentos: la necesidad de hacer al criterio simple y al mismo tiempo, se desea verlos discriminar correctamente entre buenas y malas elecciones.

Como un ejemplo de una heurística, consideremos a un gran maestro de ajedrez quién se enfrenta con la elección de varios movimientos posibles. El gran maestro decide que un movimiento en particular es el más efectivo porque ese movimiento da resultado a una posición de tablero que "parece" la más fuerte de las posiciones resultantes de otros movimientos. El criterio de "parece" más fuerte es mucho más sencilla para los grandes maestros para aplicar que digamos, la determinación rigurosa del movimiento o movimientos para el jaquemate. El hecho de que los grandes maestros no siempre ganen indica que sus heurísticas no garantizan la selección del movimiento más efectivo. Finalmente, cuando se les pregunta para que describan sus heurísticas, los grandes maestros sólo pueden dar descripciones parciales y rudimentarias de lo que a ellos les parece haber aplicado con tanto esfuerzo.

El papel de las heurísticas, ahora bien, es el de enfocar la atención sobre las *estrategias* más promisorias e intentar encontrar una solución *sin* explorar todas las posibles estratégias.

1.1.3 OPTIMIZACION, SATISFACIBILIDAD Y SEMIOPTIMIZACION

Por ejemplo, en el problema del agente viajero se requiere de un trabajo de *optimización*, la trayectoria o ruta que sea tan barata o más que cualquier otra trayectoria o ruta factible. En otros problemas de optimización, ahora bien, el objetivo no es sólo el exhibir un objeto formal que satisfaga un conjunto de criterios establecidos sino también averiguar que cualidades no marcadas posee en el espacio de candidatos. Ejemplo de este estilo es el problema de las N-reinas, (véase por ejemplo Laguna (1993)) donde se requiere de la *satisfacibilidad* de que las reinas no se maten entre sí.

La mayoría de los problemas pueden poseer ambos tipos de trabajos. Por ejemplo, el problema de las N-reinas se puede declarar como sigue: encontrar el emplazamiento más barato de las N reinas en el tablero de ajedrez donde cada reina capturada introduce un costo de una unidad. De esta manera, el problema llega a ser muy difícil puesto que no se conoce *a priori* cuando una solución de costo cero existe.

El problema del agente viajero es un ejemplo de tal situación: encontrar una ruta a través de un conjunto de ciudades es trivial en el caso de que existan todas los posibles caminos entre cualesquiera dos ciudades, pero encontrar una ruta optimal es un problema NP-completo. En tales casos, uno se ve forzado a relajar los requerimentos de optimalidad y moderar para encontrar una "buena" solución utilizando sólo un razonable esfuerzo de búsqueda. Se dice que se tiene un problema de *semioptimización* si hay un criterio de aceptación que tolera una vecindad alrededor de la solución optimal.

La mayoría de los problemas combinatorios prácticos son del tipo de semioptimización, por lo que se requiere de un balance razonable entre la calidad de la solución encontrada y el costo de búsqueda de tal solución. Más aún, dado que el esfuerzo requerido para muchos problemas de optimización combinatoria pueden alcanzar fácilmente configuraciones astronómicas, la relajación de la optimalidad es una necesidad económica. El problema básico en manejar un trabajo de semioptimización es el de idear algoritmos que garanticen cotas tanto en el esfuerzo de búsqueda como en su extensión para el cual el objetivo de optimización está comprometido. Un trabajo aún más ambicioso es el de equipar a tales algoritmos con un conjunto de parámetros ajustables que el usuario pueda cambiar para negociar entre la calidad de la solución y el monto de esfuerzo requerido.

1.1.4 PERMUTACIONES

A un arreglo ordenado de t objetos se le llama una *permutación.* Por ejemplo, existen 6 permutaciones de los tres números 1, 2, 3 son:

$$123,\ 132,\ 213,\ 231,\ 312,\ 321.$$

Las propiedades de las permutaciones son de gran importancia en el análisis de algoritmos, y se deducen muchos hechos interesantes. Inicialmente se está interesado en determinar cuantas permutaciones de n objetos son posibles: Existen n formas de elegir al elemento que está más a la izquierda, y una vez que esta elección se realiza, existen tan sólo $(n-1)$ maneras de seleccionar un objeto diferente para el siguiente lugar, esto proporciona $n(n-1)$ formas de seleccionar las dos primeras posiciones. Similarmente, se encuentra que se tienen $(n-2)$ maneras diferentes de seleccionar un elemento para la tercera posición una vez fijados los elementos de la primera y segunda posición. En general, si p_{nk} denota al número de maneras para seleccionar k objetos sacados de un conjunto de $n(n >= k)$ y se arreglan en fila, se tiene que:

$$p_{nk} = n(n-1)(n-2)\ldots(n-k+1).$$

El número total de permutaciones para los n objetos es:

$$p_{nn} = n(n-1)(n-2)\ldots 3.2.1.$$

El proceso de construir todas las permutaciones de n objetos es de una forma inductiva, suponiendo que todas las permutaciones de $(n-1)$ objetos han sido construídas, lo cual es muy importante en muchas aplicaciones.

Ahora se considerará el cómo obtener desde este arreglo las permutaciones de 4 números. Se presentarán dos métodos para ir de (n-1) a n números.

METODO 1

Para cada permutación de $a_1a_2\ldots a_{n-1}$ de $n-1$ elementos, se forman los otros n mediante insertar el número n en todos los posibles lugares, obteniendo

$$na_1a_2\ldots a_{n-1},\ a_1na_2\ldots a_{n-1},\ a_1a_2,\ldots,\ a_1a_2\ldots a_1n.$$

Por ejemplo, de la permutación 2 3 1, se obtiene, 4 2 3 1, 2 4 3 1, 2 3 4 1, 2 3 1 4. Es claro que todas las permutaciones de n objetos se obtienen de esta forma y que ninguna permutación se obtiene más de una vez.

METODO 2

Para cada permutación $a_1a_2 \ldots a_{n-1}$ de los elementos {1,2,...,n-1}, se forman los otros n de la siguiente manera: Primero constrúyase el arreglo

$$a_1a_2 \ldots a_{n-1}\frac{1}{2},\ d_1a_2 \ldots a_{n-1}\frac{3}{2},\ \ldots, a_1a_2 \ldots a_{n-1}(n-\frac{1}{2}).$$

Entonces se renombran los elementos de cada permutación usando los números 1, 2,...,n, *perservando el orden.* Por ejemplo, de la permutación 231 se obtiene

$$231\frac{1}{2},\ 231\frac{3}{2},\ :\ 231\frac{5}{2},\ 231\frac{7}{2}.$$

y renombrándolos se obtiene

$$3421,\ 3412,\ 2413,\ 2314.$$

De nuevo es claro que se obtiene cada permutación de los n elementos exáctamente una vez mediante esta construcción.

Si p_n es el número de permutaciones de n objetos i.e., $p_n = p_{nn}$, ambos métodos muestran que $p_n = np_{n-1}$.

La cantidad p_n, es llamada n *factorial* y se escribe

$$n! = (1)(2)\ldots(n) = \prod_{1\leq k\leq n} k.$$

y por convención se tiene que

$$0! = 1. \tag{1}$$

Bajo esta convención se tiene la identidad básica

$$n! = (n-1)!n \tag{2}$$

para todo entero positivo n.

Los factoriales se incrementan muy rápidamente; el número 1000! es un entero que tiene sobre 2500 dígitos decimales. Es de utilidad saber que 10! = 3,628,800, en el sentido de que "representa" una línea divisoria entre las cosas que son prácticas de calcular y las que no lo son. Así, por ejemplo, se tiene que si un algoritmo requiere de la prueba de más de 10! casos, probablemente sea en la práctica muy costoso en cuanto al tiempo de corrida.

1.2 ANALISIS DE OBJETIVOS

El análisis de objetivos (AO) es un procedimiento para desarrollar métodos de mejora para resolver problemas. Laguna y Glover (1993) comentan que una aplicación inicial fue la calendarización de la producción futura de energía de una planta nuclear, donde se descubrió la ventaja de incorporar un componente de inteligencia artificial dentro de la técnica de optimización estándar, *i.e.*, se puede considerar que el (AO) es una integración de la inteligencia artificial con la investigación de operaciones que proporciona una nueva estrategia para resolver problemas de optimización combinatoria mediante el análisis retrospectivo y técnicas de reconocimiento de patrones (*percepción tardía* Glover y Greenberg (1989)), proporcionando a los procedimientos de solución optimal o a heurísticas la habilidad para *aprender* qué reglas son las mejores para resolver una clase particular de problemas.

El análisis de objetivos proporciona un medio para explotar una "visión" amplia, junto con los beneficios de las evaluaciones de los movimientos. En particular, la incorporación de un proceso sistemático de diversificación en la búsqueda proporciona, a largo plazo, información útil para crear una función de evaluación dinámica.

Laguna y Glover (1993) proporcionan las características y pasos principales del análisis de objetivos que son:

- Fase 1.- Se aplican métodos existentes a determinadas soluciones optimales o de alta calidad a problemas representativos de una clase dada.

- Fase 2.- Se utilizan las soluciones generadas en la fase 1 como *objetivos*, los que llegan a ser el foco de un nuevo conjunto de *soluciones trasnmitidas*. Durante esa trasmisión, cada problema se vuelve a resolver, con los siguientes propósitos:

  - Evaluar la habilidad de las reglas de decisión actuales para identi-

ficar "buenos" movimientos y determinar regiones donde esas reglas están operando de manera eficiente.

- Marcar a los movimientos posibles en cada iteración para elección sesgada para seleccionar movimientos con altos puntajes.

- Generar información que pueda utilizarse para inferir los puntajes.

• Fase 3.- Construcción de funciones parametrizadas con la información registrada en la fase 2, con el objetivo de encontrar valores de los parámetros para crear una *regla de decisión maestra.* Esta regla puede también realizarse mediante el integrar funciones que complementan a la regla actual en regiones donde ésta falla para seleccionar movimientos de alto puntaje. En general, el objetivo debe ser el mejorar la habilidad general del método para realizar elecciones "correctas" en cada paso de solución.

• Fase 4.- Generación de un modelo matemático o estadístico (tal como el de la programación de metas generalizada o el modelo de análisis de discriminantes) para determinar los valores de los parámetros efectivos para la regla de decisión maestra.

• Fase 5.- Se concluye el proceso al aplicar la regla de decisión maestra a los problemas representativos originales y a otros problemas de la clase de solución elegida para confirmar su mérito.

En Glover y Greenber (1989) se proporciona el siguiente ejemplo: Suponga una estrategia de enumeración implícita que tiene una estrategia de ramificación hacia adelante donde se elige una *mejor rama* de la forma $x = b$, donde x es una elección de variable y b es una elección de su valor fijo (donde b es 0 ó 1). Suponga que un cierto criterio, digamos **k** se utiliza en la realización de la ramificación mediante la elección de los valores con pesos lineales $w_1, w_2, \ldots, w_k$. Por simplicidad, suponga que esta rama (x, b) se eligió porque:

$$\sum_i w_i F_i(x, b) = \min[\sum_i w_i F_i(\chi, \beta) : (\chi, \beta) \in B],$$

donde B es el conjunto de ramas admisibles. Después de que el problema se resuelve, la ***percepción tardía*** se utiliza para examinar las ramas que se cortarán de tal modo que se

eliminen las trayectorias muertas. La idea es ajustar los pesos w_i para asegurar al mejor ganador (conocido con percepción tardía) para los *problemas de entrenamiento.*

Para este caso las desigualdades lineales para las w_i $i = 1, 2, \ldots, k$ se generan de la siguiente manera: Sea (u, v) que denote a la rama que puede tomarse (con percepción tardía). Entonces, se ajustan los pesos para hacerlo ganar:

$$\sum_i w_i F_i(u, v) \leq \sum_i w_i F_i(\chi, \beta),$$

para toda $(\chi, \beta) \in B$.

Hasta que exista una solución a esas desigualdades, un nuevo peso se escoge. Por construcción, la elección de los pesos asegura una trayectoria más corta a una solución para todos los problemas de entrenamiento (sin trayectorias muertas). Eventualmente, puede no existir un vector de pesos que lo asegure, por lo que se requiere de algún criterio. Una forma, por supuesto, de definir una proximidad razonable es mínimos cuadrados, pero una mejor alternativa es empezar a clasificar a los problemas y obtener el mecanismo de aprendizaje de manera simultánea, así como el de tratar simultáneamente al problema para obtener los parámetros del algoritmo optimal dentro de cada clase.

1.3 ALGORITMOS GENETICOS

> "La vida empezó como el resultado de un experimento a gran escala sobre la Tierra, llamado evolución biológica. Esto empezó alrededor de cuatro mil millones de años en el Oceano primitivo en la forma de una masa viviente. De aquí, los peces se desarrollaron. Tiempo más tarde la vida se arrastró sobre la tierra. Lo siguiente, nuestros ancestros colgaron de los árboles. Y finalmente nos observamos a nosotros mismos: El hombre meditaba de cómo el método de la evolución trabaja." Rechenberg (1989).

Vale la pena reflexionar sobre la efectividad de la estrategia de la evolución biológica. En años anteriores, llegaba a ser obvio que las reglas de la evolución biológica son el resultado de un proceso de evolución en sí mismas. Suponga una población de organismos existentes con reglas hereditarias escasamente modificadas con la norma existente. Si esas modificaciones ayudan a la población para adaptarse más rápidamente a su medio ambiente particular, entonces esta población tiene mayor oportunidad de sobrevivir en

el futuro que una población con menos reglas hereditarias efectivas. Por tanto se puede asumir que la evolución, durante su acción de más de tres mil millones de años, da en sí misma un modelo optimal de operación. Al final, la estrategia de la evolución es un modo optimal que ha desarrollado fascinantes sistemas incluyendo al cerebro humano. Esto origina la idea que los conceptos estrategicos de la evolución biológica pueden utilizarse con éxito para crear máquinas "inteligentes".

1.3.1 PRIMEROS EXPERIMENTOS

Fue en 1964 cuando Rechenberg (1989) empezó con su experimento para imitar el método de la evolución biológica en un laboratorio de mecánica de fluídos. Un disco de aluminio flexible en cinco posiciones fue montado en un tunel de aire. El disco articulado podía alterarse por pasos. Existen 345,025,251 formas posibles. La tarea era el de encontrar la forma con resistencia al avance mínima. Suponiendo que no se conocía cual era, el disco era puesto en una configuración inicial aleatoria. Para producir las alternaciones aleatorias de las cinco bisagras (las mutaciones en Biología), se usaba en este primer experimento un aparato mecánico, en donde cinco pelotitas, representando a las cinco bisagras del disco, pasaban por una pirámide de puntas hacia diferentes cajas. La caja marcada determinaba el ángulo de alternación.

El arreglo experimental hace posible medir la conveniencia de las formas mutadas. La conveniencia técnica es la resistencia al avance del disco, la cual tiene que llegar a ser mínima. Al inicio del experimento del tunel de aire el disco fue plegado de una forma de zig zag de alta resistencia al avance. El esquema experimental para imitar las reglas de la evolución biológica se discutirán más adelante. La idea básica es rechazar todas las mutaciones con resistencia al avance creciente (conveniencia decreciente). Pero si una forma generada aleatoriamente tiene una resistencia al avance pequeña, entonces se tiene la forma nodriza para la siguiente generación. La resistencia al avance se midió contra el número de generaciones. Encontrando que la forma plana se tenía después de 300 generaciones.

El término técnico de *estrategia de evolución* para un programa, tiene el objetivo de copiar las reglas de la evolución biológica en la forma más exacta. Asi de lejos la imitación de la evolución de un modo simplificado ha probado ser muy exitoso. Con el tiempo se ha llegado a describir diferentes estilos de operaciones de evolución.

1.3.2 LA VENTANA DE LA EVOLUCION

La estrategia de la evolución ha probado ser de eficiencia sin precedente para muchos problemas multivariados. En este caso la longitud de paso de mutación obtiene máxima razón de progreso llegando a ser muy pequeña. Los padres y los descendientes se difunden a lo largo de la trayectoria gradiente al óptimo dentro de un pasaje muy angosto. ,De esta manera se le permite interpretar al método de la evolución biológica y a su analogía realizada por el hombre como un proceso "Hill Climbing". "Hill Climbing" es un principio común a muchas estrategias de optimización. Una estrategia "Hill Climbing" actúa como la gravedad, forzando a una pelota a rodar hacia abajo del gradiente de una montaña, pero trabaja en dirección opuesta. El efecto es que el vasto espacio de posibilidades se reduce a un angosto pasaje, en el cual la búsqueda del óptimo toma lugar. Aplicando tal estrategia se puede asegurar que una cima existe escalándola.

La existencia de la ventana de evolución proporciona el ascender a un nuevo problema. La pregunta del cómo la estrategia de evolución puede encontrar la ventana en orden a ser efectiva. Se puede responder: la estrategia de evolución multimiembro es un proceso de optimización doble. En este caso el tamaño de paso de mutación se adapta a la topología local de la función de calidad para obtener una razón máxima de progreso.

Suponga que es un alpinista escalando un lado difícil de alguna montaña. Tiene que elegir una técnica apropiada de escalamiento. Pero no puede, ahora bien, decidir si es la mejor, dado que no existe un estándar de comparación. Al día siguiente escala en grupo. Cada miembro del grupo tiene una técnica ligeramente diferente de escalamiento. Después de un corto tiempo llega a ser evidente qué técnica es mejor y se copia.

1.3.3 COMPONENTES BASICOS

Los algoritmos genéticos (AG) están basados en la noción de la propagación de nuevas soluciones a partir de *soluciones padres*, empleando mecanismos modelados de aquellos que se aplican en genética. La mejor descendencia de las soluciones padres se retiene para una nueva generación, por lo que al proceder en forma de evolución se fomenta la *supervivencia de los más aptos*. Como la calidad de los más aptos (mejor descendencia) eventualmente construye el más alto nivel compatible con el medio ambiente (el problema es gobernado por restricciones), la mejor sobre todas las descendencias se registra y es el candidato propuesto por el método para una solución optimal.

Un algoritmo genético básico tiene tres operadores: *reproducción, cruzamiento y mutación.* La reproducción es un apareamiento aleatorio de individuos (soluciones muestra) de una población para crear una o más descendencias. El cruzamiento define el resultado como un cambio de gene, cuyo valor específico es conocido como *allele*, *i.e.*, los alleles pueden concebirse como instancias en el sentido de sistemas expertos. El cambio de genes (tipo de información y sus atributos) siguen las reglas posicionales tradicionalmente modelada después de la reproducción biológica. Deben modificarse y particularizarse a diferentes tipos de problemas combinatorios, ahora bien, para que tenga sentido se debe permitiendo ciertas relaciones de restricciones para proporcionar una oportunidad al progénito de realizar un mejoramiento sobre sus padres.

Finalmente, una mutación es simplemente el introducir un elemento aleatorio, muchas veces usado para enmendar el resultado de un gene cambiado cuando el resultado no se encuentra exitoso bajo las restricciones apropiadas. A éste respecto, la mutación está más fuertemente sesgada para ayudar en la aplicación de los algoritmos genéticos que en la genética biológica. La mutación diversifica el espacio de búsqueda y protege de la pérdida de material genético que puede darse en la reproducción y el cruzamiento.

En términos generales podemos considerar que los AG son técnicas de búsqueda aleatoria que imitan a los procesos observados en la evolución natural. Combinan la supervivencia de los más aptos (o mejores) entre estructuras de anillos (soluciones) con un estructurado cambio de información aún aleatorizado.

Los algoritmos genéticos difieren de las técnicas de optimización tradicionales en muchos aspectos. Ellos trabajan con una codificación de las variables (típicamente como anillos) más que con las variables en sí mismas, y utilizan las reglas de transición probabilística para moverse de una población de soluciones a otras más que de una solución sencilla a otra. La más interesante e importante característica de los AG es que ellos utilizan tan solo evaluaciones de la función objetivo. Esto es, los AG no utilizan ninguna información sobre diferenciabilidad, convexidad o alguna otra característica auxiliar. Esto hace que los AG sean fáciles de utilizar y de implantar en gran variedad de problemas de optimización.

La aplicación de los algoritmos genéticos a problemas de investigación de operaciones se ha limitado debido a los dominios factibles complejos. Dado un problema de optimización, frecuentemente el paso más difícil en la aplicación del AG es el codificar las soluciones como anillos de forma tal que el apareamiento de soluciones factibles de por

resultado soluciones factibles.

Las técnicas para la codificación de soluciones varían de acuerdo al tipo de problema, en la mayoría de los casos, involucra cierta cantidad de arte. Por lo general para problemas enteros 0-1, las soluciones se representan típicamente por medio de una cadena de bits de longitud igual al número de variables en el problema, digamos n, donde cada posición de la cadena puede tomar el valor 0 ó 1, (en terminología genética) cada posición es un gene y el valor en esa posición es un allele. Esto resulta en un espacio de búsqueda de cardinalidad 2^n.

Existen muchas variaciones de los algoritmos genéticos debido al uso de diferentes operadores de reproducción, cruzamiento y mutación. A continuación se proporciona una estructura general de este procedimiento proporcionado por Hadj-Alouane y Bean (1993):

0.- Proporcione una generación inicial.

1.- *Copie* las N_c soluciones superiores.- Las soluciones de la generación actual son ordenadas de manera creciente respecto del valor de la función objetivo y entonces las N_c soluciones superiores se copian dentro de la siguiente generación, este procedimiento se llama reproducción.

2.-*Cruzamiento* aleatorio de parejas. Primero seleccione aleatoriamente dos cadenas (padres) de la generación actual. Ahora, cree dos descendencias como sigue: Para cada gene, una moneda sesgada se lanza, y el resultado determina cuando se realiza o no se realiza el intercambio de los alleles de los padres. Formalmente, esta operación puede describirse de la siguiente manera. Considere los anillos padres $P_1 = p_{11}p_{12}\cdots p_{1m}$ y $P_2 = p_{21}p_{22}\cdots p_{2m}$, donde p_{ij} es el allele del j-ésimo gene de la cadena P_i. Similarmente, sean $O_1 = \mathbf{o_{11}o_{12}}\ldots\mathbf{o_{1m}}$ y $O_2 = \mathbf{o_{21}o_{22}}\ldots\mathbf{o_{2m}}$ las descendencias creadas al cruzar a P_1 y a P_2. Observe que aquí, las letras en negrillas se utilizan dado que se denota al gene y no al allele. Entonces el intercambio probabilístico de los m alleles puede modelarse como m variables aleatorias independientes, digamos $X_1, X_2, \ldots, X_m$, donde cada una de ellas tiene una distribución Bernoulli con parámetros diferentes de 0.5.

Esas variables aleatorias se definen como sigue:
Para $j = 1, \ldots, m$,

$$X_j = \begin{cases} 1 & \text{si los alleles } p_{1j} \text{ y } p_{2j} \text{ se intercambian,} \\ 0 & \text{de otra forma.} \end{cases}$$

Para realizarse esos m ensayos independientes, los genes de las dos descendencias se construyen. Los genes de la descendencia son funciones de las variables aleatorias $X_j'^s$, dadas como sigue:
Para $j = 1, \ldots, m$,

$$o_{1j} = (1 - X_j)p_{1j} + X_j p_{2j},$$
$$o_{2j} = X_j p_{1j} + (1 - X_j)p_{2j}.$$

Una vez que las descendencias se crean, se evalúan, y sólo la que tiene mejor valor de la función objetivo se incluye en la nueva generación.

3.- Crear *mutación* mediante la generación aleatoria de un pequeño número de soluciones completamente nuevas e inclúyalas dentro de la nueva generación. Una solución aleatoria consiste de una cadena de m números alcatorios, digamos $r_1 r_2 \ldots r_m$ donde r_i está uniformemente distribuído en el intervalo $[1, n_i]$, para toda $i = 1, 2, \ldots, m$. Esta operación es diferente de la mutación de gene por gene dado que involucra el traer nuevos miembros a la población.

1.4 RELAJACION MATEMATICA

1.4.1 RELAJACION LAGRANGEANA

La relajación lagrangeana ha tenido gran impacto en el campo de la optimización, y representa una de las estrategias claves para resolver problemas de este dominio.

La relajación lagrangeana fue introducida por Everett (1963). La idea básica es el de sobrellevar la dificultad de trabajar directamente con un sistema de restricciones mediante el absorverlas dentro de una función objetivo a minimizarse. Esto se realiza de manera que la violación a las restricciones causará que la función objetivo sea mayor. Un intento por resolver el problema modificado (y presuntamente más fácil) con las restricciones removidas, por lo que, se tenderá hacia el descubrimiento de soluciones que satisfacen las restricciones. Dado que el problema modificado acepta soluciones como factibles que no satisfacen las restricciones absorvidas, y se emplea una función que subestima (no

puede ser mayor que) el valor de la función objetivo verdadera cuando los problemas se resuelven en la optimalidad, el problema modificado es llamado una *relajación* del original, y en general su creación y solución son el objeto del área ampliamente conocida como *relajación matemática.*

Ahora bien, el mecanismo en la relajación matemática de tomar las restricciones dentro de la función objetivo está unida a ella, en particular mediante el formar una combinación lineal no negativa (la cual multiplica cada restricción por un peso no negativo y suma el resultado). Más aún, en el caso donde las restricciones consisten de desigualdades lineales, cada variable se resume por un vector (identificando los coeficientes de las variables del problema y un lado derecho asociado). Consecuentemente, la relajación Lagrangeana se acostumbra aplicarse mediante el observar una combinación apropiada de vectores, y de manera más precisa, una mejor combinación, identificándosele como la que realiza el valor de la función objetivo relajada más grande en la optimalidad. Dado que este valor óptimo subestima al verdadero valor óptimo, una mejor combinación por tanto es la que produce la menor subestimación. (Si la subestimación puede reducirse a cero, lo cual es el resultado ideal, el problema relajado se dice que tiene una *brecha de dualidad cero.* Lo más frecuente, en problemas combinatorios, es que la subestimación no puede reducirse a cero, y se dice que el problema relajado tiene una *brecha de dualidad positiva.*)

No se conoce ninguna forma para crear la mejor combinación de vectores en un solo paso. Consecuentemente, la relajación lagrangeana procede iterativamente en pasos sucesivos, cambiando los pesos de los vectores en la función objetivo basado en el grado para el cual las restricciones asociadas son violadas o satisfechas. Si la nueva combinación de vectores es peor que la anterior, puede descartarse o combinarse con una previa, o simplemente utilizarse, como base para el siguiente estado. La mejor combinación de todas se retiene como la elección final.

1.4.2 RELAJACION DE RESTRICCIONES SUBROGADAS

La relajación de restricciones subrogadas fue introducida como una estrategia para problemas de optimización discreta por Glover (1965), participa con la relajación Lagrangeana de la noción de combinar vectores para formar combinaciones lineales no negativas.

Ahora bien, mientras que las relajaciones lagrangeanas tienen la meta de eliminar

las restricciones mediante el absorverlas dentro de la función objetivo, las relajaciones de restricciones subrogadas tienen la meta de generar nuevas restricciones que pueden ponerse expuestamente a las restricciones originales (de aquí el nombre de restricciones "subrogadas". Consecuentemente, las relajaciones de restricciones subrogadas comprometen a crear nuevos vectores con la misma forma y función de los vectores originales, y contienen adicionalmente características deseables que no se encuentran en los miembros del sistema original.

El marco de las restricciones subrogadas también introducen otro elemento que es relevante. Las restricciones formadas por las combinaciones lineales no negativas de las restricciones originales están implicadas válidamente mediante las restricciones originales, pero no son las únicas que se pueden implicar. En optimización no convexa y combinatoria, las restricciones válidas pueden también componerse mediante transformaciones para producir desigualdades llamados planos cortantes. De acuerdo a esto, las restricciones válidas derivadas de tales transformaciones proporcionan de igual manera una base para crear nuevas restricciones subrogadas. Por este aspecto las restricciones subrogadas adquieren la habilidad para operar con un sistema de vectores que es dinámico y cambiante.

Las diferencias en la perspectiva introducida por la relajación de restricciones subrogadas, en contraste con la relajación lagrangeana, tiene cierto número de consecuencias estratégicas. Una de las más notables consecuencias es el ver la creación de restricciones subrogadas como un proceso para generar información, están diseñadas para capturar información útil que no puede extraerse de las restricciones padres individualmente, pero que no son consecuencia de su conjunción.

1.4.3 DUALIDAD MATEMATICA

La dualidad matemática está intimamente relacionada con la relajación matemática mediante asociaciones que aparentemente se realizarón con el estudio de la relajación lagrangeana. La relajación lagrangeana da salida a una teoría de dualidad que es equivalente a la teoría de dualidad de Fenchel. La importancia de esta dualidad se ilustra por su habilidad de proporcionar resultados de dualidad bien conocidos en programación lineal como un caso particular.

Un problema dual lagrangeano es igual al problema de encontrar una mejor relajación lagrangeana, *i.e.*, una mejor manera para pesar a los vectores de restricciones que per-

mitan la subestimación menor del objetivo del problema original, cuando los vectores se combinan y absorven dentro del objetivo lagrangeano. De esta forma, la meta de optimizar sobre un problema dual claramente llega a ser relevante para optimizar sobre un problema primal (original).

La relajación de restricción subrogada permite una teoría de dualidad asociada pero con facetas adicionales. Un problema dual subrogado es lo mismo que encontrar una mejor relajación subrogada, como ya se vio, una mejor combinación de las restricciones originales que permitan un problema subrogado cuyo valor óptimo de la función objetivo subestime el valor óptimo del problema original mediante una pequeña cantidad.

Mediante estas caracterizaciones, se desean resultados para determinar las mejores restricciones así como el de tener resultados básicos para la teoría de dualidad.

Varios métodos constructivos han sido desarrollados y extendidos por varios autores, haciendo posible el establecer cotas para tamaños de paso con la meta de encontrar un mejor valor objetivo.

1.4.4 BUSQUEDA DISPERSA

La búsqueda dispersa (en inglés, *Scatter Search*) fue desarrollada por Fred Glover (1977), quien la caracterizó como "convenientemente ajustada para aplicaciones con estrategias de aprendizaje". La búsqueda dispersa (BD) inicia con una colección de soluciones llamadas *puntos de referencia*, que se obtienen mediante la aplicación preliminar de procesos heurísticos. En lugar de utilizar combinaciones lineales no negativas de los vectores iniciales para crear nuevos vectores, como en los marcos de la relajación Lagrangeana y la relajación subrogada, BD emplea combinaciones lineales más generales que incluyen pesos negativos.

La aproximación inicialmente genera centros pesados de gravedad, los cuales junto con subconjuntos de los puntos de referencia inicial, identifican subregiones como un fundamento para la generación de puntos secuencialmente. El permitir pesos negativos hace posible salir del área expandida por los puntos de referencia, introduciendo un elemento para incrementar la diversidad en las nuevas soluciones. Los vectores, adicionalmente se sujetan a un proceso de redondeo iterativo, para proporcionar valores enteros a los componentes que se requieren que sean discretos. Para tal proceso, una variable discreta seleccionada se redondea a un entero vecino, seguido por la determinación de los cambios

de valores implicados para las otras variables, y entonces se repite el proceso.

El propósito de modificar las componentes discretas iterativamente es para facilitar las interdependencias sobre las variables del problema. Esto proporciona un aproximación adaptativa que responde a las implicaciones de la estructura del problema y a la localización de vectores en el espacio solución.

Los nuevos vectores que resultan del proceso de combinación del proceso de BD se resometen para operarse en procedimientos heurísticos, con la meta de transformarlos dentro de soluciones de alta calidad permitiendo que el proceso empiece de nuevo.

CAPITULO 2

BUSQUEDA TABU

Muchos métodos para problemas combinatorios consisten de dos fases: construcción y mejoramiento. En este estudio se incorpora a la búsqueda tabú dentro de la fase de mejoramiento del proceso en orden a continuar la búsqueda cuando un óptimo local se encuentra. En este capítulo se da una introducción sobre el método de la búsqueda tabú y su implantación en computadora.

Cabe mencionar que no existe en la literatura un ejemplo numérico completo donde se realice un análisis de los diferentes elementos de la búsqueda tabú, así como, de la necesidad de su consideración para obtener una mayor eficiencia del proceso, en especial de las estrategias de intensificación y diversificación regional de la búsqueda y de su implantación, por lo que, con el propósito de explicar en detalle el funcionamiento de la búsqueda tabú, en este capítulo revisitamos el ejemplo numérico de Barnes y Vanston (1981), el cual es un problema de calendarización de una máquina con costos de penalización y costos de actualización de los trabajos. Cabe mencionar que en el trabajo de Barnes y Vanston, se consideran varios algoritmos de ramificación y acotamiento, así como una formulación de programación dinámica y un algotimo híbrido, este último es considerado como uno de los de mayor eficiencia para este tipo de problemas. Para el ejemplo que se presenta se utilizó la evaluación de 32 estados de este algoritmo híbrido para encontrar el valor óptimo.

Otra aspecto interesante del presente capítulo es que se presenta mediante el ejemplo numérico el concepto de "valle profundo", el cual es uno de los elementos más importantes a resolver para evitar en algunos casos el ciclado y/o la suboptimalidad.

El capítulo se desarrolla como sigue: en la sección 2.1, se plantean de manera general los problemas combinatorios; en la sección 2.2, se introduce la técnica de la búsqueda tabú estableciendo algunos conceptos y definiciones útiles describiendo los elementos básicos del algoritmo de búsqueda tabú mediante el desarrollo de algunos ejemplos. Finalmente en la sección 2.3, se discute la necesidad de la intensificación y la diversificación regional de búsqueda.

2.1 PLANTEAMIENTO DEL PROBLEMA COMBINATORIO

Considere el siguiente problema de optimización:

$$\text{(P) Minimizar } C(x) : \ x \in X \subset R^n.$$

La función objetivo puede ser lineal o no lineal, diferenciable o no, y el conjunto X es discreto.

La condición $x \in X$, sintetiza las restricciones sobre el vector x. Esas restricciones pueden incluir desigualdades lineales o no lineales. En muchas aplicaciones, puede especificar condiciones lógicas.

Algunos ejemplos clásicos que entran en este esquema son:

a). El problema del agente viajero.
b). El problema de asignación cuadrática.
c). El problema de calendarización.
d). El problema de la mochila.
e). El problema de localización.
f). El problema de agrupamiento.
g). El problema de máxima satisfacibilidad.
h). El problema de coloreado de gráficas.
i). El problema de flujo en redes.
j). El problema de reconocimiento de patrones.

Una *instancia* de un problema de optimización combinatoria puede formalizarse como una pareja (X, C), donde X denota al conjunto finito de todas las soluciones factibles y C es la función de costo, mapeo definido por

$$C : X \longrightarrow R.$$

En el caso de minimización, el problema es encontrar $i_{opt} \in X$ que satisfaga

$$C(i_{opt}) \leq C(i) \quad \forall \ i \in X.$$

A la solución i_{opt} se le llama una *solución globalmente óptima* y $C_{opt} = C(i_{opt})$ denota el costo óptimo, mientras que X_{opt} denota el conjunto de soluciones óptimas.

Por lo que un problema de *optimización combinatoria* es un conjunto digamos, I, de instancias de manera informal, una instancia está dada por los "datos de entrada" y la información suficiente para obtener una solución mientras que un problema es una colección de instancias del mismo tipo.

Existe un amplio rango de procedimientos heurísticos y optimales para resolver problemas que pueden escribirse en la forma (P). Dichos procedimientos se pueden caracterizar a través de sucesiones de movimientos, los cuales permiten pasar de un punto a otro.

Se define un movimiento s como un mapeo definido sobre un subconjunto $X(s)$ de X de la siguiente forma:

$$s : X(s) \longrightarrow X.$$

Asociado con $X(s)$ está el conjunto $S(x)$ el cual consiste de aquellos movimientos $s \in S$ que pueden aplicarse a x, por lo que se tiene la siguiente

Definición 1. *Sea (X,C) una instancia de un problema de optimización combinatoria. Entonces una estructura de vecindades es un mapeo*

$$N : X \longrightarrow 2^X,$$

que define para cada solución $i \in X$ un conjunto $S_i \subset S$ de soluciones que son "cercanas" a i en algún sentido. El conjunto S_i se llama vecindad de la solución i y cada $j \in S_i$ se llama un vecino de i. Además, se supone que $j \in S_i \iff i \in S_j$.

Es decir, se considera que un *movimiento* es una transición de una solución factible a otra solución factible (transformada), el cual se puede describir mediante un conjunto de uno o varios *atributos*. Por ejemplo, un atributo en un intercambio consiste en identificar al par de elementos que cambiaron de posición.

Este mecanismo lo utilizan diferentes métodos, en general, es utilizado por los algoritmos de optimización clásica como son los de programación lineal, programación entera, programación no lineal, etc.

2.2 BUSQUEDA TABU: UNA INTRODUCCION

La Búsqueda Tabú (BT) es un procedimiento heurístico de "alto nivel" introducido y desarrollado en su forma actual por Fred Glover (1989) y (1990a), el cual se utiliza con gran éxito para resolver problemas de optimización cuya característica principal es la de "escapar" de la optimalidad local. Para una lista actualizada de aplicaciones véase Glover y Laguna (1993).

> "La filosofía de la BT es la de manejar y explotar una colección de principios para resolver problemas de manera inteligente. Uno de los elementos fundamentales de la BT es el uso de la memoria flexible, desde el punto de vista de la BT, la memoria flexible envuelve el proceso dual de crear y explotar estructuras para tomar ventaja mediante combinar actividades de adquisición, evaluación y mejoramiento de la información de manera histórica" (Glover y Laguna (1993)).

En términos generales el método de BT puede esbozarse como:

Se desea moverse paso a paso desde una solución factible inicial de un problema de optimización combinatoria hacia una solución que proporcione el valor mínimo de la función objetivo C. Para esto se puede representar a cada solución por medio de un punto s (en algún espacio) y se define una vecindad $N(s)$ de cada punto s.

El paso básico del procedimiento consiste en empezar desde un punto factible s y generar un conjunto de soluciones en $N(s)$; entonces se escoge al mejor vecino generado $s*$ y se posiciona en ese nuevo punto ya sea que $C(s*)$ tenga o no mejor valor que $C(s)$.

Hasta este punto se está cercano a las técnicas de mejoramiento local a excepción del hecho de que se puede mover a una solución peor s^* desde s.

La característica importante de la búsqueda tabú es precisamente la construcción de una lista tabú T de movimientos: aquellos movimientos que no son permitidos (movimientos tabú) en la presente iteración. La razón de ésta lista es la de excluír los movimientos que nos pueden regresar a algún punto de una iteración anterior. Ahora bien, un movimiento permanece como tabú solo durante un cierto número de iteraciones, de forma que se tiene que T es una lista cíclica donde para cada movimiento $s \rightarrow s^*$ el movimiento opuesto $s^* \rightarrow s$ se adiciona al final de T donde el movimiento más viejo en T se elimina.

Las condiciones tabú tienen la meta de prevenir ciclos e inducir la exploración de nuevas regiones. La necesidad del significado de eliminar ciclos se debe a que, al moverse desde un óptimo local, una elección irrestricta de movimientos (persiguiendo aquellos con evaluaciones altas) permite igualmente regresarse al mismo óptimo local.

Hay que apuntar, sin embargo, que la eliminación de ciclos no es la última meta en el proceso de búsqueda. En algunos casos, una buena trayectoria de búsqueda resultará al revisitar una solución encontrada antes. El objetivo de manera amplia es el de continuar estimulando el descubrimiento de nuevas soluciones de alta calidad como se verá más adelante.

Ahora bien, las restricciones tabú no son inviolables bajo toda circunstancia. Cuando un movimiento tabú proporciona una solución mejor que cualquier otra encontrada, su clasificación tabú puede eliminarse. La condición que permite dicha eliminación se llama *criterio de aspiración.*

Es así como las restricciones tabú y el criterio de aspiración de la BT, juegan un papel dual en la restricción y guía del proceso de búsqueda. Las restricciones tabú, permiten que un movimiento se observe como admisible si no se aplican, mientras que el criterio de aspiración permite que un movimiento se observe como admisible si se satisface.

La búsqueda tabú en una forma simple descubre dos de sus elementos claves: La de restringir la búsqueda mediante la clasificación de ciertos movimientos como prohibidos (es decir, Tabú) y el de liberar la búsqueda mediante una función de memoria de término corto que proporciona una "estrategia de olvido".

Tres aspectos merecen énfasis:

(1). El uso de T proporciona la "búsqueda restringida" de elementos de la aproximación y por lo tanto las soluciones generadas dependen críticamente de la composición de T y de la manera como se actualiza.

(2). El método no hace referencia a la condición de optimalidad local, excepto implícitamente cuando un óptimo local mejora sobre la mejor solución encontrada previamente.

(3). En cada paso se elige al "mejor" movimiento.

Para problemas grandes, donde las vecindades pueden tener muchos elementos, o para problemas donde esos elementos son muy costosos para examinar, es de importancia el

áislar a un subconjunto de la vecindad, y examinar este conjunto en vez de la vecindad completa. Esto puede realizarse a pasos permitiendo al subconjunto de candidatos expanderse si los niveles de aspiración no se encuentran.

A fin de describir diferentes aspectos de BT, se consideran varios ejemplos a lo largo del capítulo y del trabajo en general.

2.2.1 EJEMPLO(PROBLEMA DE CALENDARIZACION)

Considere el problema de calendarización de una máquina con costos de penalización por retraso y costos de actualización ambos de tipo lineal.

En el tiempo cero, N trabajos llegan a una máquina de capacidad contínua. Cada trabajo i $(i = 1, 2, \ldots, N)$ requiere de t_i unidades de tiempo en la máquina y tiene una penalización de retraso por cada unidad de tiempo de p_i a partir del tiempo cero; s_{ij} es el costo de actualización de calendarizar al trabajo j inmediatamente después del trabajo i. Dos trabajos falsos 0 y $N+1$, se incluyen en cada calendario, donde $t_0 = t_{N+1} = 0$ y $p_0 = p_{N+1} = 0$. Los costos $s_{0,j}$ y $s_{i,N+1}$ se consideran como los costos de la puesta inicial y de limpieza respectivamente. Un calendario tiene la forma:

$$\pi = (0, \pi(1), \pi(2), \ldots, \pi(N), N+1),$$

donde $\pi(i)$ es el índice del trabajo en la posición i del calendario. El objetivo es el de minimizar la suma de los costos de actualización y de retraso para todos los trabajos. En términos matemáticos, se desea:

$$\text{(P)} \quad \text{Minimizar} \quad \mathrm{F}(\pi) = \mathrm{D}(\pi) + \mathrm{S}(\pi),$$

donde:

$$\mathrm{D}(\pi) = \sum_{i=1}^{N} \mathrm{d}_{\pi(i)} \mathrm{p}_{\pi(i)},$$

$$\mathrm{S}(\pi) = s_{0,\pi(1)} + \sum_{i=1}^{N-1} s_{\pi(i),\pi(i+1)} + s_{\pi(N),N+1},$$

$$\mathrm{d}_{\pi(i)} = \sum_{j=1}^{i-1} \mathrm{t}_{\pi(j)}, \quad i = 2, \ldots, N, \quad \text{y} \quad \mathrm{d}_{\pi(1)} = 0.$$

La selección preliminar de la clase de movimientos a tomarse dentro de la BT consiste en el cambio común por pares es decir, se intercambia la posición de dos trabajos

para transformar un calendario a otro. Suponga que dado un calendario el trabajo $\pi(i)$ precede, pero no necesariamente es adyacente al trabajo $\pi(j)$. Un *movimiento de intercambio* es un rearreglo de los trabajos $\pi(i)$ y $\pi(j)$ de forma tal que el trabajo $\pi(i)$ se mueve a la posición j y el tabajo $\pi(j)$ se mueve a la posición i. El *valor del movimiento* es la diferencia entre el valor de la función objetivo después del movimiento, $F(\tilde{\pi})$, y el valor de la función objetivo antes del movimiento, $F(\pi)$, *i.e.*,

$$valor\_movimiento = F(\tilde{\pi}) - F(\pi).$$

Los valores de movimiento por lo general proporcionan una base fundamental para evaluar la calidad de un movimiento, aunque otros criterios también son importantes como se verá más adelante.

Dada la solución inicial el método realiza movimientos hasta que un tiempo de computadora (*tiempo_limite*) específico transcurre. El movimiento que se realiza en cierta iteración se encuentra mediante revisar el valor de movimiento de todos los movimientos ***candidatos*** para la solución actual. Un movimiento se considera que es un candidato si los trabajos a intercambiarse están dentro de una distancia específica (número de posiciones). Dado que se está minimizando, el mejor movimiento candidato es aquel que posee el menor valor algebraico. De manera más precisa, el mejor movimiento se selecciona del conjunto de movimientos *admisibles*. Un movimiento es admisible si es no tabú o sí su estatus tabú puede eliminarse por medio del criterio de aspiración. El mejor movimiento entonces se realiza y la estructura de datos tabú se actualiza.

El proceso fundamental mediante el que BT busca trascender la optimalidad local es el de introducir un mecanismo para hacer ciertos movimientos prohibidos.

En la solución de (P), la preocupación principal es el de crear un estatus tabú que prevenga que algún movimiento se invierta bajo la juridicción de la memoria de término corto, la cual se escoge para que (P) tenga un número específico de movimientos futuros.

Es decir, la memoria de término corto de la BT constituye una forma de exploración agresiva que busca realizar el mejor movimiento posible (Esquema 2.1), sujeto a requerir elecciones posibles para satisfacer ciertas restricciones (Esquema 2.2). Esas restricciones están diseñadas para prevenir el regresarse o la repetición de cierto número de veces de ciertos movimientos mediante la ejecución de atributos seleccionados de esos movimientos prohibidos(tabú).

El Esquema 2.1, nos presentá la forma de elección del mejor movimiento admisible, esto es, dado un movimiento que es candidato, se elige al mejor de todos ellos considerando de que si es tabú debe de satisfacer el criterio de aspiración. En el Esquema 2.2, se continúa con el proceso de búsqueda hasta que un criterio de paro se satisface, éste por lo general para la técnica de la búsqueda tabú consiste de un número predeterminado de iteraciones.

Existen varias formas para crear las restricciones tabú, la Tabla 2.1 (Laguna *et.al.* (1990)) muestra una lista de posibles atributos de un intercambio de los trabajos $\pi(i)$ y $\pi(j)$ donde $j > i$, y que corresponden a restricciones que pueden imponerse para prevenir movimientos inversos.

1.- $(\pi(i), \pi(j), i, j)$	Impide cualquier movimiento que resulte en un calendario donde el trabajo $\pi(i)$ ocupe la posición i y el trabajo $\pi(j)$ ocupe la posición j.
2.- $(\pi(i), \pi(j), i, j)$	Impide cualquier movimiento que resulte en un calendario donde cualquiera de los trabajos ya sea $\pi(i)$ ocupe la posición i o el trabajo $\pi(j)$ ocupe la posición j.
3.- $(\pi(i))$	Impide a $\pi(i)$ regresarse a la posición i.
4.- $(\pi(i), i)$	Impide a $\pi(i)$ regresarse a una posición k con $k \leq i$.
5.- $\pi(i)$	Impide a $\pi(i)$ moverse.
6.- $(\pi(i), \pi(j))$	Impide a $\pi(i)$ y a $\pi(j)$ moverse.

Tabla 2.1 Restricciones tabú y atributos para movimientos de intercambio.

Otra manera de identificar atributos de un movimiento de intercambio es el de introducir información adicional, mediante no sólo hacer referencia de los elementos intercambiados, sino también de las posiciones ocupadas por esos elementos en el momento de su cambio. Se puede observar que las primeras restricciones van de menor a mayor en cuanto a que son restrictivas, pero esto no se puede afirmar de manera uniforme. Ahora bien, no existe una forma que se pueda decir que es la mejor, esto sólo se puede realizar mediante pruebas. En oocasiones es importante asegurar que las condiciones puedan manejar los procesos de solución de manera eficaz desde la vecindad actual.

La meta principal de las restricciones tabú es el permitir al método ir a puntos más allá de la optimalidad local mientras se permita la realización de movimientos de alta calidad en cada paso al mismo tiempo de que exista una negociación balanceada con respecto al esfuerzo computacional al examinar muestras muy grandes, por lo que, en oocasiones es deseable incrementar el porcentaje de movimientos posibles para que reciban una clasificación tabú. Esto se puede lograr ya sea mediante el aumento en la pertenencia tabú o mediante el cambio de la restricción tabú.

Sea π^* el calendario que proporciona el mejor valor de la función objetivo hasta el momento, entonces

Para (todos los movimientos candidatos) {

  Si (estatus del movimiento $\neq$ de tabú **ó** $F(\pi) + valor\_movimiento < F(\pi^*)$)
  {

    Si ($valor\_movimiento < mejor\_valor\_movimiento$) {

      $mejor\_valor\_movimiento \longleftarrow valor\_movimiento$;
      $mejor\_movimiento \longleftarrow$ movimiento actual;

    }

  }

}
Ejecute *mejor_movimiento*;

Esquema 2.1 Selección del mejor candidato admisible.

Además, se requiere de una estructura de datos para guardar el seguimiento de los movimientos que son clasificados como tabú y para liberar aquellos movimientos de su condición tabú cuando su pertenencia a la memoria de término corto expire. El acompañamiento de la memoria basada en la pertenencia junto con la memoria basada en la frecuencia adicionan un componente que típicamente opera sobre un horizonte. El efecto de tal memoria se puede estipular por medio de que la BT mantenga una historia selectiva H de los estados encontrados durante la búsqueda, y reemplazando la vecindad actual $N(s)$ por una vecindad modificada que depende de este proceso histórico $N(H, s)$.

El último elemento en el procedimiento básico es el *criterio del nivel de aspiración*, cuyo propósito es el de permitir que "buenos" movimientos tabú se seleccionen si el nivel de aspiración se alcanza. El apropiado uso de tal criterio puede ser muy importante para posibilitar que un método de BT alcance sus mejores niveles de realización. Este criterio de aspiración (que puede ser estándar) es el que permite que el estatus tabú se elimine si una mejor solución que la alcanzada hasta el momento se puede obtener, *i.e.*, a un movimiento tabú se le permite ejecutarse si:

$$\mathrm{F}(\pi) + \mathit{valor\_movimiento} < \mathrm{F}(\pi^*).$$

Ahora bien, otros criterios de aspiración pueden también proporcionar efectividad para mejorar la búsqueda, como se verá más adelante.

Una base para uno de esos criterios proviene de introducir el concepto de *influencia* (Glover y Laguna (1993)), la cual mide el grado de cambio inducido en la estructura de solución o de factibilidad. La influencia por lo general se asocia con la idea de distancia del movimiento, *i.e.*, donde un movimiento de gran distancia se concibe como de mayor influencia.

Genere solución inicial;

Haga {

Crear lista de candidatos { Ejecute *mejor_movimiento* };
Actualice condiciones de admisibilidad;

}
Mientras (Criterio de paro = Falso);

Termine globalmente o transfiera.

Esquema 2.2 Componente de memoria de corto término de la búsqueda tabú.

El método inicia con una solución heurística factible inicial, la cual se guarda como la mejor encontrada.

Un paso crítico, el cual envuelve la orientación agresiva de la memoria de término corto, es la elección del mejor candidato admisible. La función *mejor_movimiento* es

una que identifica a un movimiento para el cual el valor del movimiento es el más pequeño. El dominio de la función es el conjunto de todos los movimientos admisibles. El *mejor_movimiento* no tiene que ser necesariamente uno que mejore. Primero, cada uno de los movimientos de la lista de candidatos se evalua en turno.

Ahora bien, conforme la búsqueda progresa, la forma de la evaluación empleada por la búsqueda tabú llega a ser más adaptativa, incorporando referencias concernientes para la *intensificación* y la *diversificación* regional de búsqueda. Cabe aclarar que en las estrategias basadas en consideraciones de término corto la clasificación tabú sirve para identificar elementos de la vecindad del movimiento actual, mientras que en las estrategias de término intermedio y largo pueden no contener soluciones en esta vecindad, sino que por lo general consisten de seleccionar soluciones *élites* (óptimos locales de alta calidad) encontrados en varios puntos en el proceso de solución. Dichas soluciones élites se identifican como elementos de un conglomerado regional en las estrategias de intensificación de término intermedio, y como elementos de diferentes conglomerados en las estrategias de diversificación de término largo.

La esencia del método depende de cómo el registro de la historia *H* se define y se utiliza, y de cómo los candidatos y la función de evaluación se determinan.

Revisar el estatus tabú es el primer paso en la escena de la admisibilidad. Si el movimiento no es tabú, es inmediatamente aceptado como admisible; de otra forma, el criterio de aspiración da una oportunidad para eliminar el estatus tabú, proporcionando al movimiento una segunda oportunidad para clasificarse como admisible.

En algunos casos, si las restricciones tabú y el criterio de aspiración son suficientemente limitados, ninguno de los movimientos posibles, serán clasificados como admisible. Un movimiento "menos inadmisible" se salva para manipular tal posibilidad y se elige si no emergen alternativas admisibles.

La longitud de la lista tabú es un parámetro, si es demasiado pequeño el ciclado ocurrirá, pero si es demasiado grande, restringirá bastante la búsqueda para poder saltar "valles profundos" (*i.e.*, el mejor mínimo local) del espacio de valores de la función objetivo. Una faceta importante de la BT es la habilidad de localizar un rango robusto de longitudes de la lista tabú mediante pruebas empíricas preliminares para identificar para una clase de problemas los tipos de atributos y de restricciones tabú que se realizan de manera más efectiva. Acerca de esto, existe el uso de listas tabú múltiples, cada una

desarrollada para un tipo particular de atributo.

Cuando diferentes tipos de atributos se manejan de esta forma, pueden estar dados con pesos variantes, dependiendo de su clasifiación y edad, para determinar el estatus tabú de los movimientos que lo contienen.

En muchas aplicaciones, el componente de corto término por sí mismo ha producido soluciones superiores a aquellas encontradas mediante procedimientos alternativos, y el uso de memoria de mayor término en esos casos se ha eludido. Ahora bien, la memoria de término intermedio y largo puede ser importante para obtener mejores resultados para problemas difíciles, como se verá en las siguientes secciones.

La memoria de término intermedio y largo opera primariamente como una base de las estrategias de intensificar y diversificar la búsqueda.

Con el objeto de presentar el funcionamiento básico de la búsqueda tabú, se considera la siguiente instancia del problema de calendarización con costos de penalización, dado por Barnes y Vanston (1981) En el ejemplo que se presenta, la solución inicial se da de acuerdo al siguiente orden: $i < j$ implica que $(t_i/p_i) < (t_j/p_j)$.

Se conoce que la solución óptima para este problema es 13,500 para $\pi = (0, 2, 1, 4, 3, 5, 6)$, recuérdese que los trabajos 0 y 6 son sólo trabajos falsos. En este ejemplo se considerará el criterio 2 de la Tabla 2.1, *i.e.*, $(\pi(i), i, \pi(j), j)$ impide cualquier movimiento que resulte en un calendario donde cualquiera de los trabajos ya sea a $\pi(i)$ ocupe la posición i o el trabajo $\pi(j)$ ocupe la posición j, además se tienen las siguientes condiciones:

Trabajo i	Duración t_i	Penalización p_i	Radio t_i/p_i	Orden Natural
1	3	700	.004286	$\pi(5)$
2	4	800	.005	$\pi(4)$
3	1	100	.01	$\pi(3)$
4	4	300	.0133	$\pi(2)$
5	5	200	.025	$\pi(1)$

Tabla 2.2.Costos de Penalización y Actualización.

s_{ij} MATRIZ DE ACTUALIZACION DE COSTOS						
i/j	1	2	3	4	5	6
0	1100	600	1200	2000	1400	∞
1	∞	1300	700	1200	1100	1000
2	900	∞	1100	1300	600	1200
3	900	1000	∞	2000	700	1500
4	1000	700	800	∞	600	1200
5	1400	1300	1200	1300	∞	900

Tabla 2.3 Costos de Actualización.

Para iniciar nuestro proceso de búsqueda tabú consideraremos:

$$\begin{cases} \text{Punto Inicial} : \pi_0 = (0,5,4,3,2,1,6). \\ F(\pi_0) = 26600. \\ \textit{longitud\_tabu} = 7. \\ \text{Distancia máxima} = 1. \end{cases}$$

A continuación se presentarán la tablas de iteraciónes, en donde se indicará el número de iteración del proceso, los calendarios generados y los correspondientes valores de la función objetivo, así como si el calendario propuesto es un movimiento tabú y al mejor movimiento admisible para continuar con el proceso.

En este ejemplo las vecindades completas se examinan, es decir, se realizan las evaluaciones completas de los cambios de los pares hacia adelante a una distancia de uno. Entonces el mejor cambio se realiza, en este caso el que minimice la función objetivo y no sea movimiento tabú o en el caso de que lo sea para que sea admisible debe de satisfacer el criterio de aspiración que se considera que se satisface si el valor de la función objetivo mejora sobre todos los valores anteriormente encontrados.

La matriz tabú se construye al inicio del procedimiento, donde las filas de la matriz representan las posiciones y las columnas a los trabajos y se actualiza en cada iteración durante la fase de mejoramiento del algoritmo. Si un elemento (i,j) pertenece a esta matriz en una iteración dada, no se le permite realizar el cambio del trabajo i a la posición j, y se actualiza en cada iteración. Recuérdese que es posible vencer la restricción tabú en el caso de que se satisfaga el criterio de aspiración

La matriz de frecuencias es la que lleva la "historia" del procedimiento y es la que se utiliza para la formación de la función de memoria término largo, la cual permite la diversificación de la búsqueda, es decir, es posible el dirigir la búsqueda "más cercana" ó "más alejada" de las regiones exploradas.

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
0	(0,5,4,3,2,1,6)	26600		*
1	(0,4,5,3,2,1,6)	26200		*
	(0,5,3,4,2,1,6)	27300		
	(0,5,4,2,3,1,6)	26200		
	(0,5,4,3,1,2,6)	26700		

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	7		0	0	0	1	0
0	0	0	7	0		0	0	0	0	1
0	0	0	0	0		0	0	0	0	0
0	0	0	0	0		0	0	0	0	0
0	0	0	0	0		0	0	0	0	0

Observe que existen dos movimientos admisibles que nos proporcionan los mejores valores de la función objetivo, por lo que, se puede escoger cualquiera de los dos, en este caso se elige el primero *i.e.*, el movimiento que nos proporciona el calendario $\pi = (0, 4, 5, 3, 2, 1, 6)$ como punto inicial para la siguiente iteración.

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
1	(0,4,5,3,2,1,6)	26200		*
2	(0,5,4,3,2,1,6)	26600	1	
	(0,4,3,5,2,1,6)	25900		*
	(0,4,5,2,3,1,6)	26000		
	(0,4,5,3,1,2,6)	26300		

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	6		0	0	0	1	0
0	0	0	6	7		0	0	1	0	1
0	0	7	0	0		0	0	0	0	1
0	0	0	0	0		0	0	0	0	0
0	0	0	0	0		0	0	0	0	0

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
2	(0,4,3,5,2,1,6)	25900		*
3	(0,3,4,5,2,1,6)	26100	1	
	(0,4,5,3,2,1,6)	26200	1	
	(0,4,3,2,5,1,6)	22800		*
	(0,4,3,5,1,2,6)	26200		

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	5		0	0	0	1	0
0	0	0	5	6		0	0	1	0	1
0	0	6	0	7		0	1	0	0	1
0	7	0	0	0		0	0	0	0	1
0	0	0	0	0		0	0	0	0	0

Observe que en este caso el movimiento que nos proporciona el calendario $\pi = (0, 4, 3, 2, 5, 1, 6)$ es el movimiento que toma el valor objetivo más pequeño en la vecindad por lo que, se toma como punto inicial para la siguiente iteración.

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
3	(0,4,3,2,5,1,6)	22800		*
4	(0,3,4,2,5,1,6)	22800	1	
	(0,4,2,3,5,1,6)	22500	1	
	(0,4,3,5,2,1,6)	25900	1	
	(0,4,3,2,1,5,6)	19800		*

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	4		0	0	0	1	0
0	0	0	4	5		0	0	1	0	1
0	0	5	0	6		0	1	0	0	1
0	6	0	0	7		1	0	0	0	1
7	0	0	0	0		0	0	0	0	1

En esta iteración el único movimiento admisible es el que proporciona el calendario $\pi = (0, 4, 3, 2, 1, 5, 6)$ con un valor objetivo de 19800.

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
4	(0,4,3,2,1,5,6)	19800		*
5	(0,3,4,2,1,5,6)	19800	1	
	(0,4,2,3,1,5,6)	19400	1	
	(0,4,3,1,2,5,6)	19200	1	*
	(0,4,3,2,5,1,6)	22800	1	

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	3		0	0	0	1	0
0	0	0	3	4		0	0	1	0	1
0	7	4	0	5		1	1	0	0	1
7	5	0	0	6		1	1	0	0	1
6	0	0	0	0		0	0	0	0	1

Note que en esta iteración los movimientos que proporcionan los calendarios $\pi = (0,4,2,3,1,5,6)$ y $\pi = (0,4,3,1,2,5,6)$ son movimientos tabú pero ambos satisfacen el criterio de aspiración por lo que se toma como punto inicial para la siguiente iteración el calendario con valor más pequeño en la vecindad.

ITERACION	$\pi = (0,\pi(1),\pi(2),\pi(3),\pi(4),\pi(5),6)$	F(π)	TABU = 1	MEJOR Admisible
5	(0,4,3,1,2,5,6)	19200		*
6	(0,3,4,1,2,5,6)	19600	1	
	(0,4,1,3,2,5,6)	18500	1	*
	(0,4,3,2,1,5,6)	19800	1	
	(0,4,3,1,5,2,6)	23200	1	

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
0	0	0	0	2		0	0	0	1	0
0	0	7	2	3		1	0	1	0	1
7	6	3	0	4		1	1	1	0	1
6	4	0	0	5		1	1	0	0	1
5	0	0	0	0		0	0	0	0	1

En esta iteración el movimiento que proporciona el calendario $\pi = (0,4,3,2,1,5,6)$ es tabú, pero satisface el criterio de aspiración por lo que se toma como punto inicial para la siguiente iteración. De manera análoga se sigue el procedimiento hasta la iteración 11 la cual tiene las siguientes tablas:

ITERACION	$\pi = (0,\pi(1),\pi(2),\pi(3),\pi(4),\pi(5),6)$	F(π)	TABU = 1	MEJOR Admisible
10	(0,2,1,4,3,5,6)	13500		***
11	(0,1,2,4,3,5,6)	14100	1	
	(0,2,4,1,3,5,6)	15500	1	
	(0,2,1,3,4,5,6)	14000	1	
	(0,2,1,4,5,3,6)	14700		*

MATRIZ TABU						MATRIZ DE FRECUENCIAS				
6	0	0	3	0		1	1	0	1	0
3	6	2	5	0		2	1	1	1	1
2	5	4	0	0		1	2	1	1	1
1	4	7	0	0		1	1	1	0	2
0	0	0	0	7		0	0	1	0	1

Dado que se conoce de antemáno la solución optimal del problema, el cual se obtiene en la iteración 10 y se señala por un triple asterisco, sólo se presentan estas iteraciones. En general, si en alguna iteración ya no existen puntos admisibles y no se ha satisfecho el criterio de paro, se tendría que utilizar las funciones de memoria de término intermedio (intensificación) y de término largo (diversificación), que se verán en la siguiente sección.

El contador de frecuencia muestra la distribución de los movimientos a través de las iteraciones. Se utiliza ese contador para diversificar la búsqueda, maniobrando dentro de nuevas regiones. Esta influencia de diversificación se restringe para operarse sólo en ocasiones particulares. En este caso, donde ningún movimiento de mejora admisible existe. El uso de la información de la frecuencia se utiliza por lo general para penalizar movimientos que no mejoran mediante el asignar una penalización grande a los pares intercambiados con mayor frecuencia provocando con esto que se pierda lo atractivo de tales intercambios.

En suma, las frecuencias definidas sobre diferentes subconjuntos de soluciones anteriores, particularmente subconjuntos de soluciones élites consistentes de óptimos locales de alta calidad encontrados en varios puntos en el proceso de solución, proporcionan las estrategias complementarias de intensificación.

Una aproximación que está cercanamente unida a los orígenes de la BT y que proporciona un interjuego efectivo entre la intensificación y la diversificación es la *estrategia de oscilación.*

2.2.2 ESTRATEGIA DE OSCILACION

La estrategia de oscilación opera mediante el moverse hasta pegarle a una frontera, representada por la factibilidad o un estado de construcción que normalmente puede

representarse por un punto donde el método puede parar. En vez de parar, ahora bien, la definición de vecindad se extiende o el criterio de evaluación para seleccionar movimientos se modifica, para permitir que se pueda cruzar esa frontera. La aproximación entonces procede para una profundidad específica más allá de la frontera y entonces se regresa. En este punto la frontera de nuevo se aproxima y se cruza, esta vez desde la dirección opuesta, procediendo a un nuevo punto en turno. El proceso de aproximar y cruzar repetidamente la frontera desde diferentes direcciones crea una forma de oscilación que es la que le da el nombre a la estrategia. El control sobre esta oscilación se establece mediante el generar evaluaciones y reglas de movimientos modificadas, dependiendo de la región en la cual se está actualmente navegando y de la dirección de la búsqueda.

Un ejemplo simple de esta aproximación ocurre para el problema de la mochila multidimensional donde los valores de las variables cero-uno se cambian de 0 a 1 hasta alcanzar la frontera de factibilidad. El método entonces continua dentro de la región infactible utilizando el mismo tipo de cambios, pero con un evaluador modificado. Después de un número seleccionado de pasos, la dirección se invierte mediante el cambiar las variables de 1 a 0. El criterio de evaluación se maneja para mejorar y varía de acuerdo a cuando el movimiento es de más a menos infactible o de menos a más infactible y se acompañan mediante restricciones asociadas sobre cambios admisibles para los valores de las variables.

Ahora bien, para incorporar la estrategia de oscilación, no necesariamente se tiene que definir en términos de factibiliad, sino que puede definirse donde la búsqueda parece gravitar. La oscilación consiste en forzar la búsqueda a movimientos fuera de esta región y el permitir regresarse a la región, ofreciendo de esta manera una forma efectiva para eliminar entrampamientos suboptimales en las búsquedas estándares.

2.3 MEMORIA DE TERMINO INTERMEDIO Y LARGO

Como se ha visto, el método de la BT empieza con una solución factible inicial y en el proceso de ejecución, el procedimiento actualiza a los arreglos y elementos de la función memoria. Entonces el proceso se repite hasta que el criterio de terminación se encuentra.

En el método de BT descrito en el ejemplo, el "mejor" movimiento que se realiza en cada iteración se especifica como el movimiento admisible con el menor valor objetivo. Ahora bien, esta estrategia no garantiza que el movimiento seleccionado permita la búsqueda en la dirección de la solución optimal, por lo que, se requiere de técnicas

que nos permitan integrar las estrategias de intensificación y diversificación de manera efectiva, basándose sobre las funciones de memoria de término intermedio y largo de la BT. En otras palabras, es de importancia vital el "mirar" la *dependencia regional* de buenos criterios de decisión, no sólo en términos de movimientos de mejoramiento y no mejoramiento.

La memoria de término intermedio opera para registrar y comparar características de las mejores soluciones generadas durante un periodo particular de búsqueda. Las características que son comunes o que competen a la mayoría de esas soluciones se toman como atributo regional. El método entonces procura nuevas soluciones que tengan esas características regionales.

La función de memoria de término largo, emplea principios que son inversos a los de la función de término intermedio. La memoria de término largo se utiliza para producir un punto inicial de búsqueda en una nueva región, mediante el penalizar las características que la memoria de término intermedio encuentra que prevalecen en la región actual de búsqueda.

2.3.1 INTENSIFICACION Y DIVERSIFICACION

La fase de intensificación proporciona una forma simple para enfocar la búsqueda alrededor de la mejor solución (o conjunto de soluciones élites) hasta el momento.

Para entender la importancia de estos recursos de la BT, consideraremos dos corridas del ejemplo numérico considerado en la sección anterior pero variando los valores de algunos de los parámetros iniciales.

CORRIDA 1

$$\begin{cases} \text{Punto Inicial} : \pi_0 = (0,5,4,3,2,1,6). \\ F(\pi_0) = 26600. \\ \textit{longitud\_tabu} = 7. \\ \text{Distancia máxima} = 2. \end{cases}$$

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
0	(0,5,4,3,2,1,6)	26600		*
1	(0,5,2,3,4,1,6)	25500		*
	(0,3,4,5,2,1,6)	26100		
	(0,5,4,1,2,3,6)	26600		
2	(0,3,2,5,4,1,6)	20700		*
	(0,5,4,3,2,1,6)	26600	1	
	(0,5,2,1,4,3,6)	22600		

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
3	(0,5,2,3,4,1,6)	25500	1	
	(0,3,4,5,2,1,6)	26100	1	
	(0,3,2,1,4,5,6)	14900		*
4	(0,1,2,3,4,5,6)	14900	1	
	(0,3,4,1,2,5,6)	19600	1	
	(0,3,2,5,4,1,6)	20700	1	

Como se puede observar, en la iteración 3 se llegó a un óptimo local, el cual está dado por el calendario $\pi = (0, 3, 2, 1, 4, 5, 6)$ con un valor objetivo de 14900, y en la iteración 4 también se llega a otro óptimo local dado por el calendario $\pi = (0, 1, 2, 3, 4, 5, 6)$ con igual valor objetivo, pero en éste último caso se tiene que el punto es tabú y además no se tienen soluciones admisibles. Se puede entonces pensar en escoger a éste último óptimo local para intensificar la búsqueda dentro de la región por lo que se toma como punto de arranque y se inicializa la tabla tabú para continuar la búsqueda, pero se observa que en la iteración 10 se cae en un ciclo, y no existe mejoramiento por lo que se puede considerar que se está en un valle profundo.

Ahora bien, en este ejemplo la búsqueda pudo estar muy restringida debido al valor de la longitud tabú que es muy alto, por lo que se realizó una segunda corrida:

CORRIDA 2

Punto Inicial : $\pi_0 = (0,5,4,3,2,1,6)$
$F(\pi_0) = 26600$
longitud_tabu = 3
Distancia máxima = 2

ITERACION	$\pi = (0,\pi(1),\pi(2),\pi(3),\pi(4),\pi(5),6)$	$F(\pi)$	TABU = 1	MEJOR Admisible
0	(0,5,4,3,2,1,6)	26600		*
1	(0,5,2,3,4,1,6) (0,3,4,5,2,1,6) (0,5,4,1,2,3,6)	25500 26100 26600		*
2	(0,3,2,5,4,1,6) (0,5,4,3,2,1,6) (0,5,2,1,4,3,6)	20700 26600 22600	 1 	*

ITERACION	$\pi = (0,\pi(1),\pi(2),\pi(3),\pi(4),\pi(5),6)$	$F(\pi)$	TABU = 1	MEJOR Admisible
3	(0,5,2,3,4,1,6) (0,3,4,5,2,1,6) (0,3,2,1,4,5,6)	25500 26100 14900	1 1 	 *
4	(0,1,2,3,4,5,6) (0,3,4,1,2,5,6) (0,3,2,5,4,1,6)	14900 19600 20700	1 1	 *

Se puede observar que en la iteración 3 se alcanza un óptimo local y en la iteración 4 se alcanza el otro óptimo local que constituyen un agujero negro, pero en esta corrida a diferencia de la anterior se tiene un punto admisible el cual se toma para proseguir con la búsqueda pero en las iteraciónes 13 y 14 se vuelven a alcanzar los óptimos locales y se inicia un ciclo a partir de la iteración 20.

Las dos corridas anteriores indican la necesidad de contar con elementos que nos permitan salir de este tipo de entrampamientos suboptimales, por lo que se debe de

recurrir a un análisis retrospectivo del proceso de búsqueda que nos pueda proporcionar un reconocimiento de patrones de comportamiento para poder identificar regiones no visitadas.

A manera de ejemplo se considera la matriz de frecuencias de la corrida 2 después de 20 iteraciones, donde las filas indican las posiciones y las columnas los trabajos:

2	0	3	0	2
0	4	0	3	0
4	0	4	0	5
0	3	0	4	0
2	0	2	0	2

Tabla 2.4 Matriz de Frecuencias de la Corrida 2.

Una forma de "aprender" qué regiones no se han visitado, es el observar la frecuencia con la cual un cierto trabajo no se ha seleccionado para una posición particular, así por ejemplo, se tiene que el trabajo 1 no se ha localizado en las posiciones 2 y 4, que el trabajo 3 se ha localizado más frecuentemente en la posición 3 de los calendarios, etcétera.

A partir de la matriz histórica de la búsqueda se pueden generar calendarios que se utilizan como puntos de arranque para nuevos procesos de búsqueda, a manera de ilustración se tienen los siguientes ejemplos. Seleccionando el calendario inicial de forma empírica a partir de la Tabla 2.4, tomando los trabajos en la posición donde no se han localizado (donde aparecen los ceros) o donde se ha localizado el menor número de veces.

$$\begin{cases} \text{Punto Inicial} : \pi_0 = (0,2,1,5,3,4,6). \\ F(\pi_0) = 16300. \\ \textit{longitud\_tabu} = 3. \\ \text{Distancia máxima} = 2. \end{cases}$$

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
0	(0,2,1,5,3,4,6)	16300		*
1	(0,2,1,4,3,5,6)	13500		*
	(0,5,1,2,3,4,6)	23400		
	(0,2,3,5,1,4,6)	18500		

$$\begin{cases} \text{Punto Inicial} : \pi_0 = (0,2,3,5,1,4,6). \\ \text{F}(\pi_0) = 18500. \\ \textit{longitud\_tabu} = 3. \\ \text{Distancia máxima} = 2. \end{cases}$$

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
0	(0,2,3,5,1,4,6)	18500		*
1	(0,2,1,5,3,4,6)	16300		*
	(0,5,3,2,1,4,6)	23100		
	(0,2,3,4,1,5,6)	17300		
2	(0,5,1,2,3,4,6)	23400		
	(0,2,3,5,1,4,6)	18500	1	
	(0,2,1,4,3,5,6)	13500		*

$$\begin{cases} \text{Punto Inicial} : \pi_0 = (0,5,1,4,3,2,6). \\ \text{F}(\pi_0) = 24500. \\ \textit{longitud\_tabu} = 3. \\ \text{Distancia máxima} = 2. \end{cases}$$

ITERACION	$\pi = (0, \pi(1), \pi(2), \pi(3), \pi(4), \pi(5), 6)$	F(π)	TABU = 1	MEJOR Admisible
0	(0,5,1,4,3,2,6)	24500		*
1	(0,4,1,5,3,2,6)	23300		*
	(0,5,3,4,1,2,6)	27800		
	(0,5,1,2,3,4,6)	23400		
2	(0,5,1,4,3,2,6)	24500	1	
	(0,4,3,5,1,2,6)	26200		
	(0,4;1,2,3,5,6)	18900		*
3	(0,2,1,4,3,5,6)	13500	1	*
	(0,4,3,2,1,5,6)	19800		
	(0,4,1,5,3,2,6)	23300	1	

Ahora bien, la diversificación se restringe para operarse sólo en ocasiones particulares. En este caso, se seleccionan las ocasiones donde ningún movimiento de mejora admisible existe. Por lo general, se utiliza la información de la frecuencia para penalizar a los movimientos que no mejoran la búsqueda mediante el asignar una penalización grande a intercambios de pares con mayores contadores de frecuencia.

Una implantación sencilla de tal técnica se puede realizar de la siguiente manera: Primero, se cuenta el número de veces que cada movimiento digamos m se ha realizado en orden a calcular su frecuencia $f(m)$. Entonces una penalización $p(m)$ se asocia a cada movimiento de la siguiente manera:

$$p(m) = \begin{cases} 0 & \text{si } m \text{ alcanza un criterio de aspiración.} \\ wf(m) & \text{de otra forma.} \end{cases}$$

donde w es una constante. Entonces el valor del movimiento puede ser:

$$F(x+m) - F(x) + p(m).$$

El peso w depende del problema, del tipo de movimiento y de la vecindad. Glover y Laguna (1993) indican que en muchas aplicaciones se ha observado que este peso es aproximadamente proporcional a la raíz cuadrada del tamaño de la vecindad multiplicada por la desviación estándar del valor (sin penalización) de cada movimiento ejecutado durante la búsqueda.

Como se puede observar, la función de penalización depende directamente del criterio de aspiración, por lo que ahora nos enfocaremos a este concepto.

2.3.2 CRITERIOS DE ASPIRACION

Los criterios de aspiración se introducen en la BT para determinar cuando las restricciones tabú pueden sobrellevarse para remover una clasificación tabú que de otra manera se aplicaría a un movimiento. El uso apropiado de tales criterios puede ser muy importante para que un método BT alcance sus mejores niveles de realización.

En las primeras aplicaciones de la BT se aplicó tan sólo un tipo sencillo de criterio de aspiración, consistente de remover una clasificación tabú a un movimiento si éste permitía una solución mejor que la mejor encontrada hasta el momento, tal regla se ilustró en los ejemplos de los capítulos anteriores.

Siguiendo a Glover y Laguna (1993), las aspiraciones son de dos clases: *aspiraciones de movimiento* y *aspiraciones de atributo*. Una aspiración de movimiento, cuando se satisface, revoca la clasificación tabú del movimiento. Una aspiración de atributo, cuando se satisface revoca el estatus tabú del atributo. En éste último caso el movimiento puede o no cambiar su clasificación tabú, dependiendo de sí la restricción tabú puede activarse por más de un atributo. Así entonces se pueden tener los siguientes criterios de aspiración:

Aspiración por Default: Si todos los movimientos posibles son clasificados como tabú, entonces el movimiento "menos tabú" se selecciona.

Aspiración por Objetivo.: Una aspiración de movimiento se satisface, permitiendo que un movimiento x sea un candidato para seleccionarse si $F(x) <$ mejor costo.

Aspiración por Dirección de Búsqueda: Un atributo de aspiración para e se satisface si la dirección en e proporciona un mejoramiento y el actual movimiento es un movimiento de mejora.

CAPITULO 3

ANALISIS DE LOCALIZACION

En el campo de la optimización combinatoria surgen una gran variedad de problemas prácticos y teóricos de manera extensa e interesante, algunos de ellos fáciles de resolver, pero otros con un alto grado de dificultad.

En este capítulo se presenta la bondad de la técnica de la búsqueda tabú para El Problema de Asignación Cuadrático, así como los lineamientos a seguir para implantarlo en computadora. El trabajo presenta el desarrollo de algoritmos de la técnica de la búsqueda tabú, que a diferencia de los reportados en la literatura no utilizan ninguna rutina de construcción ni de mejoramiento inicial antes de aplicar la búsqueda tabú. A diferencia de los resultados reportados en la literatura, nuestros métodos encuentran los valores óptimos o mejores valores reportados en la literatura sin recurrir a la fase de diversificación regional de búsqueda, en la mayoría de los casos estudiados.

El problema de asignación cuadrático es parte del tema general de la *elección optimal dentro de un contexto espacial.* Ejemplos de tales elecciones incluyen la localización de fábricas, de almacenes, de escuelas, de servicios de urgencias. Tales decisiones típicamente se han basado sobre consideraciones de tipo económico, social, militar, político o del medio ambiente.

En los problemas de localización, generalmente las soluciones factibles, elección de sitios de localización, vienen dadas por las posibles permutaciones de los elementos, es por ello que las técnicas de búsqueda tabú desarrolladas para El Problema de Asignación Cuadrático pueden ser fácilmente modificadas para utilizarse en la solución de los problemas que se describen en la sección 3.1.

El capítulo se desarrolla como sigue: en la sección 3.1, a manera de introducción se describen: el problema p-mediana, el problema p-centrado y el problema de localización de plantas sin capacidad, en la sección 3.2, se describe y analiza el problema de asignación cuadrático así como la implantación del método de búsqueda tabú para su solución, se describen algunos criterios de nivel de aspiración, así como algunos métodos del manejo de la lista tabú de tipo dinámico. Finalmente, en la sección 3.3, se hace un análisis comparativo de tres desarrollos propuestos del método tabú y se reporta la experiencia

computacional.

3.1 PROBLEMAS DE LOCALIZACION

La literatura al respecto es muy amplia, y existen muchas formulaciones y clasificaciones como se puede ver en de los Cobos Silva (1987) y Love *et. al.* (1988), para una revisión amplia de aplicaciones y formulaciones del problema de localización se puede ver Brandeau y Chiu (1989).

3.1.1 PROBLEMAS p-MEDIANA

Considérese a la red $N = (V, A)$ consistente del conjunto de vértices V y un conjunto de arcos A. Con cada arco que va del vértice v_a al vértice v_b y denotado como $[v_a, v_b]$ está asociado un número real positivo $\alpha(v_a, v_b)$ llamado la *longitud* de arco. Entonces, para un par de nodos en una red, inicialmente se está interesado en encontrar una de las trayectorias más cortas *i.e.*, una trayectoria de longitud mínima que conecte a esos dos nodos, la cual se denotará por $P[v_a, v_b]$ con una longitud $d(v_a, v_b)$. La matriz simétrica asociada a las distancias entre los vértices se denotará por D.

Supóngase ahora que cada uno de los nodos de la red corresponde a un cliente y que el i-ésimo cliente tiene una demanda de w_i unidades de cierto bien por periodo de tiempo, la cual es independiente de la localización de la planta que lo satisfaga. Más aún, supóngase que cada nodo es un sitio de localización potencial para una planta la cual puede producir cualquier cantidad del bien deseado (por unidad de tiempo) y enviarlo a través de los caminos representados por los arcos. También supóngase que cada entrega del bien requiere un viaje sencillo entre una planta y un cliente, *i.e.*, no se consideran los escenarios donde un viaje satisface varias demandas. A menos de que exista ambigüedad, usaremos cliente, f, y planta, g, como equivalente con "un cliente localizado en o representado por el nodo v_f" y "una planta puesta en la ubicación v_g".

Las plantas pueden estar *abiertas* o *cerradas*. La planta i se dice cerrada si ninguna planta es localizada en la ubicación v_i y por tanto ningún cliente puede ser servido desde esa ubicación potencial, de otra forma se dirá que la planta está abierta, y a menos de que se diga lo contrario, las plantas abiertas son *sin capacidad* en el sentido de que pueden satisfacer a cualquier número de clientes.

El problema p-mediana (P-pM) se establece de la siguiente manera: dado un número p

entero positivo ($p \leq |V|$), se desea establecer p plantas en p de las ubicaciones potenciales tal que las demandas de todos los clientes sean satisfechas de manera que el costo total en el que se incurra sea mínimo.

Entonces lo que se desea es encontrar una solución optimal, la que se obtiene resolviendo:

$$\text{Minimizar}_{y,z} \sum_{k \in K} \sum_{j=1}^{n} c_{kj} y_{kj},$$

sujeto a

$$\sum_{k \in K} y_{kj} = 1 \quad \text{para toda } j,$$

$$z_k - y_{kj} \geq 0 \quad \text{para todas } k \in K, j,$$

$$\sum_{k \in K} z_k = p,$$

$$z_k \in \{0,1\}, \quad y_{kj} \geq 0 \text{ para todas } \quad k \in K, j.$$

donde:

K es el conjunto de vértices donde se pueden localizar las plantas (K subconjunto de V).

$c_{kj} = w_k d(v_k, v_j)$ es el costo total en el que se incurre si todas las w_k unidades demandadas por el cliente k son enviadas por la planta j.

y_{kj} es la proporción de la demanda w_j de la planta j existente, asignada a una nueva planta localizada en k.

$z_k = 1$, si la ubicación k se elige para una nueva planta y cero de otra forma.

El primer conjunto de restricciones asegura que la demanda de cada planta existente es completamente distribuída. El segundo conjunto de restricciones asegura de que no existe distribución en la ubicación k a menos de que una planta se encuentre abierta. La restricción final requiere que exáctamente p plantas estén abiertas.

3.1.2 PROBLEMAS p-CENTRADOS

Sea la red $N = (V, A)$ y sea $X_p = \{x_1, x_2, \ldots, x_p\}$ el conjunto de p puntos, esto es, X_p subconjunto de V y $D(y, X_p) = \min_{x \in X_p} d(y, x)$, donde $d(x, y)$ es el valor de la trayectoria más cortas entre x y y. Por tanto el problema p-centrado se formula como:

$$H(X_p) = \min \max_{y \in V} D(y, X_p),$$

donde V es el conjunto de las m posibles ubicaciones para las plantas.

El problema p-centrado P-pC, difiere significativamente del P-pM en varios aspectos, principalmente con respecto al criterio usado para calcular la calidad de una solución factible. Mientras que el P-pM es un problema *minisum*, el P-pC tiene un objetivo *minimax*: Abiertas p plantas y asignando a cada cliente a exáctamente una de ellas tal que la *distancia máxima* de cualquier planta abierta a cualquiera de los clientes asignados a ella sea mínima.

Las funciones objetivo minimax ocurren frecuentemente en formulaciones de problemas de localización para servicios de emergencia tales como estaciones de policía, de bomberos o de servicios de ambulancia.

3.1.3 PROBLEMAS DE LOCALIZACION DE FACILIDAD SIN CAPACIDAD

Sea $J = \{1, \ldots, n\}$ un conjunto finito de n posibles lugares para el establecimiento de nuevas plantas o para redimensionar las ya existentes. El problema de localización de planta sin capacidad (PLFSC) trata del abastecimiento de un bien desde un subconjunto de esas ubicaciones potenciales para ser plantas a un conjunto $I = \{1, \ldots, m\}$ de m clientes con demandas preestablecidas del bien. Las plantas se suponen de capacidad ilimitada de tal manera que cualquier planta puede satisfacer toda la demanda. Existen costos asociados con las plantas y con las rutas de transportación de los sitios potenciales como plantas hacia los clientes, lo que se desea es encontrar un plan de producción-transportación de costo mínimo que satisfaga toda la demanda o equivalentemente que maximice la utilidad.

Sea f_j el costo fijo por abrir la planta j y que existe una utilidad conocida c_{ij} que se tiene cuando se satisface al cliente i desde la ubicación j. Típicamente, c_{ij} es una función de los costos de producción en la planta j, los costos de transportación desde la

planta j al cliente i, la demanda del cliente i y el precio de venta para el cliente i. Por ejemplo, $c_{ij} = d_i(p_i - q_j - t_{ij})$ donde d_i es la demanda, p_i el precio por unidad, q_j el costo de producción por unidad y t_{ij} el costo de transporte por unidad.

Para cualquier conjunto S de plantas abiertas, es óptimo servir al cliente i desde la planta j para el cual es c_{ij} es máximo sobre $j \in S$. Entonces, dado S la utilidad es: .

$$z(S) = \sum_{i \in I} \max_{j \in S} c_{ij} - \sum_{j \in S} f_j.$$

El problema por tanto es, encontrar un conjunto S que permita la máxima utilidad Z, *i.e.*, $Z = \max_{S \subset J} z(S)$. Esta formulación es como un problema combinatorio.

Ahora bien, una formulación como programa lineal entero, se obtiene mediante el introducir las variables $x_j = 1$, si la planta j está abierta y cero de otra forma, y a la variable $y_{ij} = 1$, si la demanda del cliente i se satisface desde la planta j y cero de otra forma. Por lo que el programa lineal entero queda como:

$$Z = \max \sum_{i \in I} \sum_{j \in J} c_{ij} - \sum_{j \in J} f_j x_j$$

sujeto a:

$$\sum_{j \in J} y_{ij} = 1, \quad \text{para toda } i \in I,$$

$$y_{ij} \leq x_j, \quad \text{para toda } i \in I,\ j \in J,$$

$$y_{ij}, x_j \in \{0,1\}, \quad \text{para toda } i \in I,\ j \in J.$$

3.2 EL PROBLEMA DE ASIGNACION CUADRATICO

En su forma más simple, el problema de asignación cuadrático (de aquí en adelante se abreviará QAP) puede describirse como sigue. Encontrar la asignación óptima de n plantas a n ubicaciones para minimizar el producto acumulado de flujo entre cualesquiera dos plantas por las distancias entre cualesquiera dos ubicaciones. De manera más precisa, si f_{ip} es el flujo entre las plantas i y p, y d_{jq} es la distancia entre las ubicaciones j y q, el problema puede escribirse como:

$$\text{Minimizar} \sum_i \sum_p \sum_j \sum_q f_{ip} d_{jq} x_{ij} x_{pq},$$

sujeto a:

$$\sum_j x_{ij} = 1 \,\forall i,\ \sum_i x_{ij} = 1 \,\forall j,\quad x_{ij} = 0, 1.$$

La variable x_{ij} es diferente de cero si el objetivo i es asignado a la ubicación j y es cero de otra forma. La función a optimizarse es cuadrática y no convexa, i.e., existe cierto número de óptimos locales, además el conjunto factible es un conjunto de permutaciones.

Una forma sencilla de interpretar el problema planteado es suponer que existen n sitios disponibles y se van a construir n edificios en estos lugares, entonces si f_{ip} es el número de gentes por unidad de tiempo que viajarán entre los edificios i y p, y d_{jq} es la distancia entre los sitios j y q, lo que se desea es encontrar la asignación de los sitios de construcción de manera que la distancia total recorrida se minimice.

Un aspecto interesante de este problema es el hecho de que cuando una de las matrices de la función objetivo se restringe a ser una matriz de permutaciones cíclica, el problema de asignación cuadrático se reduce al problema del agente viajero.

Una formulación equivalente del problema es:

$$\text{Minimizar}\quad F(\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\pi(i)\pi(j)},$$

donde π es una permutación del conjunto N=$\{1,2,...,n\}$. Equivalentemente, se desea encontrar una permutación π del conjuntó N que minimice el valor de $F(\bullet)$.

3.2.1 ELEMENTOS DE LA BT PARA EL QAP

Los siguientes definiciones describen los elementos de la búsqueda tabú para el QAP. Un movimiento es una transición de una permutación a otra. Un atributo de un movimiento en este caso es simplemente un par no ordenado (i, j) de plantas que cambian sus ubicaciones. El valor de un movimiento es la diferencia entre los valores de la función objetivo después y antes del movimiento. Si este valor es negativo, el movimiento proporciona una mejora. La función *mejor_movimiento* es una que identifica al par (i_{mejor}, j_{mejor}) para el cual el valor de movimiento es el más pequeño. El dominio de la función *mejor_movimiento* es el conjunto de movimientos admisibles. El mejor movimiento no tiene que ser necesariamente uno que mejore.

Una lista tabú de pares (i, j) de longitud *longitud_tabú* se construye y actualiza. Si un par (i, j) pertenece a la lista tabú para una iteración dada, no se permite el intercambio

de las plantas i y j en esa iteración, a menos que se satisfaga el criterio de aspiración que en este caso se cumple si el intercambio de las plantas i y j proporciona un valor de la función objetivo estrictamente mejor que cualquiera de los obtenidos hasta el momento.

3.2.2 ESTRATEGIA BASICA

La estrategia básica que se utilizará está dividida en cuatro fases: fase de inicialización, fase de mejoramiento, fase de intensificación y fase de diversificación.

Cada iteración de la fase de mejoramiento evalúa todos los posibles movimientos y determina el mejor movimiento valuado admisible. Si ninguna solución mejora después de cierto número prefijado de iteraciones, por ejemplo, $k1 \times n$, donde $k1$ es número natural, la fase de intensificación inicia.

La fase de intensificación inicia, con una lista tabú vacía, desde la mejor solución encontrada en la región actual. Si después de $k2 \times n$ iteraciones con $k2$ número natural no se ha encontrado una mejor solución, la fase de diversificación inicia. El algoritmo termina después de que la búsqueda se ha diversificado a un número dado $k3$ de regiones.

Este proceso se encuentra expresado en pseudocódigo en el algoritmo 3.1.

La **Fase 1** nos da la inicialización del algoritmo al encontrar una solución inicial.

En la **Fase 2** se comienza a generar nuevas soluciones a partir de la actual vía la estructura de vecindades. La Fase 2a genera todas las soluciones vecinas a la actual y selecciona la mejor. En la Fase 2b se toma como solución actual a la mejor solución encontrada y se actualizan la matriz tabú y la matriz de frecuencias.

La Fase 2c pregunta si ha habido mejoramiento en el valor objetivo de la solución actual en menos de $k1 \times n$ iteraciones. Si la respuesta es afirmativa se va nuevamente al paso (a) de la Fase 2, en caso contrario se va a la Fase 3.

El objetivo de la Fase 2 del Algoritmo 3.1 es encontrar una solución de calidad a partir de una solución dada. Esta fase termina identificando la mejor solución encontrada en una región de factibilidad en la que se encuentra la solución inicial dada.

ALGORITMO 3.1

Fase 1 *Inicialización:* Genere solución inicial y vaya a la Fase 2.

Fase 2 *Mejoramiento:*

a) Evalúe todos los movimientos, determine y realice el mejor.
b) Actualice.
c) Si no hay mejoramiento en $k1 \times n$ iteraciones vaya a Fase 3.
De otra forma vaya a Fase 2.

Fase 3 *Intensificación:*

a) Vaya a la mejor solución encontrada e inicialice la lista tabú.
b) Evalúe todos los movimientos, determine y realice el mejor.
i) Actualice.
ii) Si no hay mejoramiento en $k2 \times n$ iteraciones:
iii) Si la Fase 3 ha encontrado una mejor solución,
vaya a Fase 3a.
De otra forma:
iv) Si $k3$ reinicios se han realizado, pare.
De otra forma vaya a Fase 4.
De otra forma vaya a Fase 3b.

Fase 4 *Diversificación:*

a) Determine las restricciones tabú de la memoria de término largo.
b) Evalúe todos los movimientos, determine y realice el mejor.
i) Actualice.
Si no hay mejoramiento en $k3$ iteraciones:
ii) Inicialice con la mejor solución.
iii) Elimine las restricciones tabú adicionales debidas a la
memoria de largo término, actualice el número de
reinicios y vaya a la Fase 2.
De otra forma vaya a Fase 4b.

La Fase 3 inicia con la mejor solución encontrada hasta el momento, si es que no ha habido mejoramiento en las últimas $k1 \times n$ iteraciones en el proceso de mejoramiento y se inicializa la tabla tabú. Se procede de manera análoga a la fase de mejoramiento durante $k2 \times n$ iteraciones, con la diferencia de que, si se encuentra una mejor solución se regresa a la Fase 3a siempre y cuando no se haya alcanzado un número de reinicios, en cuyo caso el algoritmo para. En el caso en que no se encuentren mejores soluciones durante $k2 \times n$ y no se alcance el número de reinicios, se pasa a la Fase 4.

La Fase 4 primero determina las restricciones tabú de la memoria de término largo como se verá posteriormente. Determinadas las restricciones en la Fase 4a se se encuentra una nueva solución y se genera su estructura de vecindades. La Fase 4b evalúa a todos las soluciones vecinas, seleccionando a la mejor y actualizando a la lista tabú y a la matriz de frecuencias. Si no hay mejoramiento en $k3$ iteraciones, entonces se inicializa con la mejor solución encontrada, se eliminan las restricciones tabú adicionales, se actualiza el número de reinicios y se pasa a la Fase 2, en caso contrario, se continúa con la Fase 4b.

Para la diversificación se utiliza la frecuencia basada en la memoria de término largo. Esta memoria de término largo puede representarse por medio de una matriz LT de nxn donde cada elemento LT_{ij} almacena el número de veces en que cada entrada de una permutación toma el valor de 1 en todas las soluciones examinadas por la búsqueda, de esta manera, si un cierto porcentaje de las entradas con mayor frecuencia, se guardan como tabú para un número suficiente de iteraciones, la búsqueda se forzará para la exploración de nuevas regiones. Alternativamentè, una nueva solución inicial puede construírse basada en la matriz de término largo.

Ahora bien, la diversificación se restringe para operarse sólo en ocasiones particulares. En este caso, se seleccionan las ocasiones donde ningún movimiento de mejora admisible existe. Por lo general, se utiliza la información de la frecuencia para penalizar a los movimientos que no mejoran la búsqueda mediante el asignar una penalización grande a intercambios de pares con mayores contadores de frecuencia.

Una implantación sencilla de tal técnica se puede realizar de la siguiente manera: Primero, se cuenta el número de veces que cada movimiento digamos m se ha realizado en orden a calcular su frecuencia $f(m)$. Entonces una penalización $p(m)$ se asocia a cada movimiento de la siguiente manera:

$$p(m) = \begin{cases} 0 & \text{si } m \text{ alcanza un criterio de aspiración.} \\ wf(m) & \text{de otra forma.} \end{cases}$$

donde w es una constante. Entonces el valor del movimiento puede ser:

$$F(x+m) - F(x) + p(m).$$

El peso w depende del problema, del tipo de movimiento y de la vecindad como ya se ha mencionado antes.

Como se puede observar, la función de penalización depende directamente del criterio de aspiración, por lo que a continuación se proponen algunos de éstos

Una vez que se han identificado los movimientos factibles y se ha calculado su correspondiente valor de la función objetivo, el estatus tabú de los candidatos factibles se prueba. Suponga que en una iteración dada (*iter*) el valor de $\pi(s)$, donde $s = s_i$, cambia de i a i', entonces a la planta $\pi(i)$ le está prohibido ocupar la posición i durante un número específico de iteraciones. La matriz *tiempo_tabú* se utiliza para forzar esta restricción tabú. El elemento (i, $\pi(i)$) de la matriz *tiempo_tabú* contiene el número de iteración en la cual se le permite de nuevo a la planta $\pi(i)$ ser asignada a la posición i. Si el valor de la función objetivo de la solución anterior al movimiento es mejor que el valor de la función objetivo después de éste, entonces el tiempo tabú se calcula como:

$$tiempo\_tabú(\text{i}, \pi(i)) = 0.25\ n^2,$$

de otra forma:

$$tiempo\_tabú(\text{i}, \pi(i)) = 0.75\ n^2.$$

Este es un criterio de aspiración implícito, donde a los elementos que contribuyen a soluciones con "buenos" valores de la función objetivo se les permite regresar más rápidamente para la prueba de soluciones actuales. El máximo tiempo que un movimiento puede clasificarse como tabú es el equivalente al 75% del tamaño de la estructura de *tiempo_tabú*. Esto significa que en una iteración dada un mínimo del 25% de los movimientos posibles no se clasifican como tabú.

Ahora bien un criterio de nivel de aspiración de tipo explícito es el siguiente, el estatus tabú de un movimiento se elimina si el movimiento permite la búsqueda a una

nueva solución *mejor de la región.* La solución mejor de la región es aquella con el mejor valor de la función objetivo durante digamos las n^2 últimas iteraciones, se consideró el valor de n^2 puesto que no es ni muy pequeño como n ni demasiado grande. Observe que este nivel de aspiración es más flexible que el utilizado típicamente en las implantaciones de BT, donde la mejor solución sobre todas sirve para propósitos similares.

Otro criterio explícito es una aspiración por *dirección del movimiento.* para implantar este criterio es necesario construir una matriz *dirección_tabú.* Si el movimiento (i,$\pi(i)$) llega a ser tabú durante una fase de mejoramiento, entonces el elemento (i,$\pi(i)$) de la matriz dirección_tabú contiene el número de la iteración de cuando la fase de mejoramiento comenzó, de otra forma al elemento se le da el valor de cero. El movimiento candidato factible (i,$\pi(i)$) es elegible para seleccionarse en la iteración *iter* si:

F(movimiento posterior) $\neq$ F(movimiento anterior) y ocurre al menos una de las siguientes:

$$\begin{cases} -tiempo\_tabú(i,\pi(i)) < iter. \\ -F(movimientoposterior)\ ¡F(mejordelaregión). \\ -dirección\_tabú(i,\pi(i)) \neq 0 \text{ y } dirección\_tabú(i,\pi(i)) = fase\_actual. \end{cases}$$

donde F(•) es el valor de la función objetivo y *fase_actual* es cero si la búsqueda se encuentra en una fase de no mejoramiento o el número de la iteración donde la actual fase de mejoramiento comenzó. El mejor movimiento en cualquier iteración es el movimiento elegible con mejor valor de movimiento, para mayor detalle véase Laguna y Kelly (1993).

3.2.3 MANEJO DINAMICO DE LA LISTA TABU

El *manejo de la lista tabú* significa la actualización de ésta, *i.e.*, la decisión de cuántos y cuáles movimientos se pondrán como tabú para una iteración de la búsqueda. La mayoría de las implantaciones de la búsqueda tabú utilizan un manejo de la lista tabú de tipo estático. De manera más precisa, los movimientos permanecen como tabú durante un número de iteraciones fijo, por lo que la eficiencia del algoritmo depende de la elección de la duración del estatus tabú o, equivalentemente, de la longitud de la lista tabú.

Este manejo de la lista tabú de tipo dinámico permite el exámen más detallado de la región factible, por lo que es posible romper ciclos y diversificar la búsqueda.

Existen varios métodos de lista tabú dinámica, entre los que destacan: el método de

la *secuencia de cancelación* (CSM) y el método de *eliminación inversa* (REM) propuestos en Glover (1990a), y que en el capítulo 4 se explican de manera más detallada.

Ahora bien, también se puede manejar una longitud de LT de manera dinámica, donde la longitud de la lista tabú varía de manera dinámica a través de diferentes configuraciones pasando de una a otra si ningún mejoramiento se encuentra en un número dado, de iteraciones. Este conjunto de configuraciones permite un exámen más detallado de la región factible mediante el incremento y decremento del número de movimientos tabú actuales vía las configuraciones. Además se propone un componente aleatorio que actúa cada vez que la lista tabú cae en la configuración 1 como un elemento de seguridad adicional contra el ciclado, para mayores detalles refiérase a Chakrapani y Skorin-Kapov (1993)

3.3 EXPERIENCIA COMPUTACIONAL

Los procesos usados para la experimentación fueron implementados en lenguaje C en una PC con tarjeta 486 DLC a 40 Mhz.

Varias versiones de BT y que se describen a continuación se probarón sobre el conjunto estándar de problemas de tamaño n = 5, 6, 7, 8, 12, 15, 20, 30 dados por Nugent *et. al.* (1967), los cuales se utilizan actualmente de manera amplia para probar la eficiencia de los algoritmos.

Para todo el conjunto de problemas se encontraron las mejores soluciones reportadas en la literatura.

Las versiones que se utilizaron de la búsqueda tabú tienen las siguientes características:

BT1 : Estructura básica de BT, *i.e.*, se consideró la longitud de la lista tabú como un parámetro constante, el criterio de nivel de aspiración fue el estándar de la literatura, es decir, el nivel de aspiración se satisface si el movimiento a realizarse mejora el valor de la función objetivo con respecto de todos los movimientos realizados.

BT2 : Se consideró una estructura básica de la BT y se agregó el criterio de nivel de aspiración proporcionado por la matriz *tiempo_tabú* dado en la sección anterior.

BT3 : En este marco de la BT se consideró el criterio de nivel de aspiración estándar, así como, un manejo de la lista tabú de tipo dinámico, parecido al propuesto por Chakrapani y Skorin-Kapov (1993) que utilizaron para programación en paralelo.

3.3.1 MANEJO DINAMICO PROPUESTO

Si no existe mejoramiento, entonces el movimiento se considera tabú si:

$$liminf \leq tabu(i,j) \leq limsup,$$

donde:

$$liminf = \begin{cases} 1 & numiter \text{ Mod } 8 < 5, \\ ltabu(numiter \text{ Mod } 8)(0.2) & \text{de otra forma.} \end{cases}$$

$$limsup = \begin{cases} ltabu((numiter \text{ Mod } 8) - 4)(0.2) & \text{si } numiter \text{ Mod } 8 < 5, \\ ltabu & \text{de otra forma.} \end{cases}$$

donde *numiter* es el número de iteración, *ltabu* es la longitud tabú y "Mod" es la función módulo.

En todos los casos, para la fase de diversificación regional de la búsqueda (aunque sólo se llegó a utilizar en una de las instancias de tamaño n=30.), se consideró una función de penalización $p(\bullet)$ como la descrita anteriormente, la cual no se ha reportado en la literatura para su adaptación en el problema de asignación cuadrático.

En las Tablas 3.2 y 3.3 se presentan entre paréntesis los mejores valores encontrados, así como, el número de iteraciones en las que se alcanzó, antes de llegar a un número máximo de iteraciones impuesto *max_iter*, el cual dependía del tamaño del problema y se proporciona en la Tabla 3.1. Los valores con asterisco son los valores óptimos.

n	5	6	7	8	12	15	20	30
max_iter	15	15	25	25	700	2500	7500	12000

Tabla 3.1

Los resultados obtenidos fueron los siguientes:

n	Pto. Inic.	Valor Inic.	Long. Tabú	BT1		BT2		BT3		Valor Mejor
				Valor	Iter.	Valor	Iter.	Valor	Iter.	
5	1	66	5	(58)	1	50	11	50	14	50*
	2	58	5	(58)	0	50	11	50	11	50*
	3	72	5	50	1	50	1	50	1	50*
	4	68	5	(52)	6	50	12	50	4	50*
	5	82	5	(52)	5	50	12	50	12	50*
6	1	86	4	86	0	86	0	86	0	86*
	2	110	4	86	2	86	2	86	2	86*
	3	108	4	86	6	92	1	86	6	86*
	4	116	4	86	1	86	1	86	1	86*
	5	116	4	86	8	86	11	86	8	86*
7	1	174	5	148	23	148	21	148	23	148*
	2	240	5	148	10	148	10	148	10	148*
	3	216	5	148	4	148	4	148	4	148*
	4	228	5	148	18	148	18	148	18	148*
	5	202	5	148	19	148	19	148	19	148*
8	1	214	6	214	10	214	16	214	10	214*
	2	322	6	214	5	214	5	214	5	214*
	3	290	6	214	7	214	15	214	7	214*
	4	320	6	214	5	214	5	214	5	214*
	5	288	6	214	4	214	4	214	4	214*
12	1	784	9	(586)	57	578	683	578	368	578*
	2	784	9	578	377	(586)	110	578	18	578*
	3	874	9	578	12	578	16	578	11	578*
	4	850	9	(586)	23	578	84	578	53	578*
	5	746	9	578	5	578	5	578	4	578*
15	1	1448	10	1150	1495	1150	46	1150	200	1150*
	2	1612	10	1150	1468	1150	375	1150	231	1150*
	3	1596	10	1150	2368	(1152)	53	(1152)	51	1150*
	4	1610	10	1150	1124	(1152)	181	(1152)	305	1150*
	5	1626	10	(1152)	33	(1152)	32	(1152)	32	1150*

Tabla 3.2 Valores de la función objetivo y número de iteraciones.

n	Pto. Inic.	Valor Inic.	Long. Tabú	BT1		BT2		BT3		Valor Mejor
				Valor	Iter.	Valor	Iter.	Valor	Iter.	
20	1	3444	12	2580	7435	2570	559	2570	1014	2570
	2	3302	12	2570	5626	2570	954	2570	1841	2570
	3	3540	12	2570	1869	2570	701	2570	2197	2570
	4	3456	12	2570	918	2570	890	2570	887	2570
	5	3322	12	2570	1191	2570	613	2570	2408	2570
30	1	8060	18	6124	11048	6124	3628	(6158)	171	6124
	2	7758	18	6124	7294	(6128)	6223	6124	10692	6124
	3	8172	18	(6128)	972	6124	9842	(6148)	7740	6124
	4	7648	18	(6128)	2713	6124	3995	(6128)	1049	6124
	5	8224	18	6124	5748	6124	3839	(6148)	7881	6124

Tabla 3.3 Valores de la función objetivo y número de iteraciones.

Cabe mencionar que la versión BT2 presentada en el trabajo no se ha reportado en la literatura como adaptada para el problema de asignación cuadrático.

Los puntos iniciales que se utilizaron para las corridas son los puntos iniciales dados por Nugent. Cabe mencionar que ésta es una gran diferencia con respecto a los reportes en la literatura, donde antes de utilizar a la búsqueda tabú, utilizan algunas rutinas de construcción inicial, las cuales proporcionan "buenos" puntos iniciales.

A continuación se presentan: la tabla 3.4 que proporciona los promedios de los valores y del número de iteraciones de las corridas para cada uno de los diferentes tamaños de problemas, la tabla 3.5 que proporciona las eficiencias de los promedios de los valores encontrados con respecto de los mejores valores reportados en la literatura, finalmente, la tabla 3.6 indica en su parte izquierda el número promedio de iteraciones para los casos en que se encontrarón los valores óptimos o los mejores reportados, y en la parte derecha de la tabla están los cocientes del número de iteraciones de cada uno de los algoritmos, indicándonos la proporción de iteraciones de cada uno de éstos.

n	BT1		BT2		BT3		Valor
	Valor	Iter.	Valor	Iter.	Valor	Iter.	Mejor
5	54	2.6	50	9.4	50	8.4	50*
6	86	3.4	87.2	3	86	3.4	86*
7	148	14.8	148	14.4	148	14.8	148*
8	214	6.2	214	9	214	6.2	214*
12	581.2	94.8	579.6	179.6	578	90.8	578*
15	1150.4	1297.6	1151.2	137.4	1151.2	163.8	1150*
20	2570	3407.8	2570	743.4	2570	1669.4	2570
30	6125.6	5555	6124.8	5505.4	6141.2	5506.6	6124

Tabla 3.4 Promedios de las corridas.

La eficiencia se calculó tomando como base el mejor valor reportado z_{mej}, y se usó la ecuación:

$$eficiencia = 1 - \frac{z - z_{mej}}{z_{mej}}.$$

Tabla de eficiencias sobre valores promedios.			
5	0.920	1	1
6	1	0.986	1
7	1	1	1
8	1	1	1
12	0.994	0.997	1
15	0.999	0.998	0.998
20	1	1	1
30	0.999	1	0.997

Tabla 3.5

No se puede observar que alguno de los algoritmos sea consistentemente más eficiente. En la Tabla 3.6, se presentá el promedio de iteraciones para cuando se alcanzaron los mejores valores reportados, así como, los porcentajes de su número dc iteraciones.

Tabla del promedio de iteraciones.

n	Promedio de Iteraciones BT1	BT2	BT3	Porcentajes BT1	BT2	BT3
5	-	9.4	8.4	-	1.119	1
6	3.4	3	3.4	1.133	1	1.133
7	14.8	14.4	14.8	1.027	1	1.027
8	6.2	9	6.2	1	1.451	1
12	131.33	197	90.8	1.446	2.169	1
15	1613.75	210.5	215.5	7.666	1	1.023
20	3407.8	743.4	1669.4	4.584	1	2.245
30	8030	5326	10692	1.507	1	2.007

Tabla 3.6

Se puede observar que el método BT2 proporciona en la mayoría de los casos el menor número de iteraciones para alcanzar el mejor valor, siendo en algunos casos una relación de más de 7 a 1 en el némero de iteraciones.

CAPITULO 4

INVENTARIOS MULTIPRODUCTOS

En la actualidad son grandes los retos a los que se debe de enfrentar las empresas, entre los que destacan la competencia creciente, las dificultades de abastecimiento, la sobreabundancia de artículos y la escaces de materia prima, por lo que es de importancia cada día mayor la adecuada administración económica de los inventarios.

Una forma de asegurar la continuidad de las operaciones es el almacenamiento, sin embargo, esto desencadena costos suplementarios, lo que por efecto ocaciona una reducción del margen de utilidad, por otro lado, la falta de inventarios adecuados puede propiciar un rompimiento en la continuidad de operación y altos costos de oportunidad por no poder satisfacer la demanda.

En este capítulo se desarrolló para caso de los los modelos de inventario multiproducto con demanda determinística y con costos de reaprovisionamiento conjunto, un algoritmo realmente original, no tan sólo porque es la primera vez que se utiliza un esquema de la búsqueda tabú para este tipo de problemas, sino que además proporciona la mayor eficiencia en cuanto a las soluciones obtenidas con respecto a los algoritmos reportados como de los más eficientes. Se reporta en la experiencia computacional, la eficiencia de este nuevo marco de solución propuesto como la mayor en todas las instancias consideradas, con respecto de los demás algoritmos probados.

El capítulo se desarrolla como sigue: en la Sección 4.1, se describe el modelo de inventarios multiproducto para el caso de demanda determinística, así como algunos de los algorítmos que resulven el problema con eficiencia; en la Sección 4.2, se considera a la búsqueda tabú como un metaprocedimiento para realizar una fase de mejoramiento y se realiza un análisis comparativo contra otros métodos, en la sección 4.3, se describen algunos métodos de manejo de la lista tabú de tipo dinámico. Finalmente en la Sección 4.4, se presenta la experiencia computacional.

4.1 INVENTARIOS MULTIPRODUCTO: CASO DETERMINISTICO

El problema de inventarios multiproductos consiste en general de determinar cuánto y cuándo ordenar de cada artículo de los que se controlan, para que, la demanda sea

satisfecha con costos míminos. En este trabajo se supone que la entrega por parte de los proveedores es inmediata y se realiza en una sola entrega, no se permite déficit y los artículos nunca llegan a ser obsoletos una vez que se inventarean. Además se supone que diferentes artículos de un mismo proveedor se pueden entregar en un solo paquete y se realiza a intervalos regulares de tiempo, digamos t_1.

El problema para n artículos por tanto se traduce en determinar la frecuencia de los ciclos de abasto tanto individual como conjunto, por lo que se puede observar fácilmente que el cociente de la frecuencia individual del artículo j digamos t_j entre la frecuencia conjunta es un entero positivo β_j mayor o igual a uno, i.e., $t_j = \beta_j t_1,\ j = 1, 2, \ldots, n$

Los artículos tienen diferentes ciclos, pero el que marca las oportunidades para ordenar es el menor duración t_1. Se ordena tantas veces como lo requiera dicho artículo en una unidad de tiempo, $1/t_j$, y cada vez que se ordena se incluyen en la orden los artículos cuyo ciclo así lo indica.

Los costos que intervienen en el modelo son de: ordenación, adquisición e inventario. Un costo fijo K se carga cada vez que se realiza una orden. Un costo por línea k_j se produce cada vez que se carga el artículo j en la orden correspondiente.

El costo debido al inventario se carga sobre el inventario promedio durante un ciclo y está dado por:

$$\frac{1}{2}\sum_{j=1}^{n} h_j d_j t_j.$$

donde: h_j, d_j, y t_j son el inventario promedio, la demanda promedio y el tiempo del ciclo del artículo j, respectivamente.

De lo anterior se tiene que el costo total por unidad de tiempo es:

$$CT = \frac{1}{t_1}(K + \sum_{j=1}^{n} \frac{k_j}{\beta_j} + \sum_{j=1}^{n} c_j d_j) + \frac{1}{2} t_1 \sum_{j=1}^{n} h_j d_j \beta_j.$$

el cual se desea minimizar. El costo total no es sólo función de t_1, sino también de las$\beta_j, j = 1, 2, \ldots, n$ y donde las c_j, $j = 1, 2, \ldots, n$ son los costos unitarios.

La solución propuesta en Narro Ramírez (1993) consiste en aproximar los valores de las $\beta_j'^s$ y después encontrar el valor de t_1 que minimiza a $CT(t_1)$, de la siguiente manera:

El costo correspondiente a cada artículo ordenado se puede expresar como: $\alpha_j K$, con$\alpha_j \geq 0$, $\sum_{j=1}^{n} \alpha_j = 1$. Si la política óptima para cada artículo j, considerado como independiente, aconseja ordenar x_j unidades en cierto instante, entonces se tiene que:

$$K\delta(\sum_{j=1}^{n} x_j) + \sum_{j=1}^{n} k_j \delta(x_j) \geq \sum_{j=1}^{n} (\alpha_j K + k_j)\delta(x_j),$$

donde $\delta(x) = 1$ si $x > 0$, y cero de otra forma.

La política óptima es una política factible para el sistema de inventario del artículo j con costo por ordenar $(\alpha_j K + k_j)$. Si el costo mínimo correspondiente es $(CT)_j(\alpha_j)$, una cota inferior para el costo de la política óptima completa está dada por: $\sum_{j=1}^{n}(CT)_j(\alpha_j)$, para cualquier asignación de las α'^s, por lo que la mejor cota inferior está dada por:

$$\text{Minimizar}_{\alpha_i} \quad \sum_i (CT)_i(\alpha_i)$$

$$\text{s.a.} \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, n, \quad \sum_{i=1}^{n} \alpha_i = 1.$$

Ahora bien, las formas en que se calculan los ciclos y se aproximan los múltiplos son diversas. Si l es el real que resulta del cociente del tiempo de duración del ciclo correspondiente y el de menor longitud y β es el entero más próximo, entonces se debe satisfacer que: $\beta(\beta+1) < l^2 < (\beta+1)l$.

El algorítmo propuesto por Narro Ramírez (1993) se resume en:

Paso 1.- Resolver:

$$\text{Minimizar}_{\alpha_i} \quad \sum_i (CT)_i(\alpha_i)$$

$$\text{s.a.} \quad \alpha_i \geq 0, \quad i = 1, 2, \ldots, n, \quad \sum_{i=1}^{n} \alpha_i = 1.$$

Paso 2.- Calcular para j=1,2,...,n:

$$l_j'^2 = \frac{k_j}{d_j h_j} \max\left\{\frac{d_i h_i}{\mathrm{K} + k_i}, i = 1, 2, \ldots, n\right\},$$

$$\beta_j'^2 = \frac{1+\sqrt{1+4l_j'^2}}{2},$$

$$t_1' = \sqrt{\frac{2(\mathrm{K}+\sum_{i=1}^{n}\frac{k_i}{\beta_i'})}{\sum_{i=1}^{n} h_i d_i \beta_i'}},$$

$$l_j^2 = \frac{2(\alpha_j \mathrm{K} + k_j)}{\frac{d_j h_j}{t_1'^2}},$$

$$\beta_j^2 = \frac{1+\sqrt{1+4l_j^2}}{2}.$$

paso 3.- Resolver:

$$\text{Minimizar} \frac{1}{t_1}(\mathrm{K} + \sum_{j=1}^{n}\frac{k_j}{\beta_j} + \sum_{j=1}^{n} c_j d_j) + \frac{1}{2} t_1 \sum_{j=1}^{n} h_j d_j \beta_j \quad \text{s.a. } t_1 \geq 0.$$

4.2 MARCO DE MEJORAMIENTO POR BUSQUEDA TABU.

A fin de iniciar la búsqueda para mejorar los planes maestros de los pedidos a proveedores una orden inicial se requiere. Esta se obtiene de manera natural al resolver el paso 1 y poner $\beta_j = 1, j = 1, 2, \ldots, n$. Ahora bien, para utilizar a la búsqueda tabú como un metaprocedimiento de mejora, en lugar de utilizar valores fijos para las β_j, $j = 1, 2, \ldots, n$, se propone que cada β esté dentro de un rango de valores dado por:

$$\beta_j \in [1, \beta_j^* + 1], \text{ donde } \beta_j^* = \left\lceil \sqrt{\frac{1+\sqrt{1+4l_j^2}}{2}} \right\rceil \text{ para } j = 1, 2, \ldots, n,$$

donde $\lceil x \rceil$ es el entero mayor más cercano de x.

Por tanto, lo que se desea es resolver:

$$\text{Minimizar} \frac{1}{t_1}(\mathrm{K} + \sum_{j=1}^{n} \frac{k_j}{\beta_j} + \sum_{j=1}^{n} c_j d_j) + \frac{1}{2} t_1 \sum_{j=1}^{n} h_j d_j \beta_j,$$

$$\text{s.a.} \quad t_1 \geq 0, \quad \beta_j^* \in [1, \beta_j^* + 1], \quad j = 1, 2, \ldots, n.$$

el cual es un problema de programación no lineal entero mixto, pero como se observa la función objetivo resulta ser una función convexa en t_1, por lo que, una vez fijos los valores de las β'^s, el valor óptimo para t_1 se encuentra fácilmente.

Considerarémos que una solución es de la forma:

$$\pi = (\pi(1), \pi(2), \ldots, \pi(n)) \text{ donde } \pi(i) \in [1, \beta_i^* + 1], \ i = 1, 2, \ldots, n.$$

entonces, definimos un movimiento de la forma $\pi' = \pi \pm e_j, \quad j = 1, 2, \ldots, n,$ tal que $\pi'(i) \in [1, \beta_i^* + 1], \ i = 1, 2, \ldots, n.$ En una implantación sencilla de BT, la lista tabú se mantiene en forma implícita registrando sólo el índice j de la variable entera cuyo valor ha cambiado en la iteración.

4.3 MANEJO DINAMICO DE LA LISTA TABU

El *manejo de la lista tabú* significa la actualización de ésta, *i.e.*, la decisión de cuántos y cuáles movimientos se pondrán como tabú para una iteración de la búsqueda. La mayoría de las implantaciones de la búsqueda tabú utilizan un manejo de la lista tabú de tipo estático. De manera más precisa, los movimientos permanecen como tabú durante un número de iteraciones, por lo que la eficiencia del algoritmo depende de la elección de la duración del estatus tabú o, equivalentemente, de la longitud de esta lista.

Este manejo de la lista tabú de tipo dinámico permite el exámen más detallado de la región factible, por lo que es posible romper ciclos y diversificar la búsqueda.

Existen varios métodos de lista tabú dinámica, entre los que destacan: el método de la *secuencia de Cancelación* (CSM) y el método de *eliminación inversa* (REM) propuestos en Glover (1990a), los que a continuación se presentan.

Denotemos por LT a la lista tabú que implícitamente define al conjunto de movimientos tabú. Como punto inicial, considérese a LT dada de la siguiente manera: $LT =$

$(e(1), e(2), ..., e(q))$. Supongamos que los elementos de LT son indexados por la iteración, identificando el punto en el cual fue adicionado a la lista, y q es el índice de la iteración actual. Correspondientemente se puede identificar a la lista de soluciones: $(x(1), x(2), ..., x(q))$ tal que, para cada i, $e(i)$ es el atributo asociado con el movimiento aplicado a $x(i)$.

Ahora bien, en general se supondrá que los atributos de movimientos satisfacen una propiedad de *suficiencia*, la cual especifica que ninguna de dos soluciones, $x(h)$ y $x(k)$, para $h < k$, pueden ser la misma a menos de que exista un interjuego de los elementos $e(h), \ldots, e(k)$ tal que, para cadà par $e(r)$, $e(s)$ del conjunto se tenga que, $\tilde{e}(r) = e(s)$, donde $\tilde{e}(\bullet)$ es el atributo inverso o complementario de $e(\bullet)$. Es de utilidad el mantener esta propiedad basada sobre un proceso de "cancelación sucesiva", es decir ningún atributo e puede aparecer dos veces sobre LT a menos de que $\tilde{e}$ aparezca en una posición intermedia.

En la sucesión $e(p), \ldots, e(q)$ si cualquier $e(r)$ es seguido por un elemento tal que $e(r) = \tilde{e}(s)$, entonces se dice que $e(r)$ está *cancelado* por el primero de tales $e(s)$ (*i.e.*, $e(s)$ está indexado por el último $s > r$).

Se asocia con LT, una *lista tabú activa*, LTA, la cual consiste sólo de los elementos de LT que no han sido cancelados, por lo que LTA es una subsucesión de LT que contiene el mismo último elemento y que se puede describir como:

$$LTA = (e(p), \ldots, e(h), e(i), e(j), \ldots, e(q), e(q+1)).$$

Si la adición de $e(q+1)$ constituye la primera vez que algún elemento cancela a un elemento previo, entonces por la propiedad de suficiencia se tiene que ninguna de cualesquiera dos soluciones generadas alejadamente pueden ser la misma. Más aún, la solución $x(i)$ no puede duplicarse por la solución de cualquier sucesión de movimientos futuros a menos de que cada elemento de la sucesión $e(j), \ldots, e(q)$ esté cancelado.

A esta sucesión que cae entre el cancelamiento del elemento $e(q+1)$ y el elemento cancelado $e(i)$ se le llama la *sucesión de cancelación* o *C-sucesión*. Dado que se previene la cancelación de $e(q)$ por $e(q+1)$, se puede asegurar que el uítimo elemento en cada *C-sucesión* sucesiva permanecerá sin cancelar, por lo que ninguna solución puede repetirse. Esto es una condición suficiente pero no necesaria para eliminar repetición de soluciones.

El segundo proceso dinámico para manejar listas tabú está basado en la generación de restricciones de movimientos implicadas de manera general por las *C-sucesiones* con

cotas que se pueden tanto expander como contraer. Lo apropiado de expander las cotas de una *C-sucesión* proviene del hecho de que la adición de un nuevo elemento a la *LTA* proteje contra el regreso a una solución "guardada" por otras *C-succsiones* y así pueda incluirse entre esas sucesiones. Existe un obstáculo para realizar esto dado que el nuevo elemento es no adyacente a los elementos previos de las *C-sucesiones* y por lo general no existe manera de rearreglar a los elementos y a las cotas, tal que, esas *C-sucesiones*, se puedan definir por los tipos convenientes de estructuras de datos que se introducen para manejar a las *C-sucesiones* ordinarias.

El marco para sobrepasar este obstáculo opera mediante la eliminación sucesiva de los elementos de la lista *LT* por una secuencia inversa y no por una secuencia para mantener a *LTA* actualizada en cada iteración, por lo que el método se denomina de *eliminación inversa*. El estatus tabú asignado por REM representa un criterio necesario y suficiente para prevenir el revisitar soluciones conocidas, como se explica en Glover (1990a) y en Dammeyer y Voβ (1993).

4.3.1 METODO DE ELIMINACION INVERSA

Sin un control adicional, un proceso de búsqueda puede revisitar en la siguiente iteración una solución óptima local al moverse en una vecindad de ésta. Considere que los atributos de todos los movimientos se almacenan en una *lista de corridas*, *i.e.*, una lista que representa las trayectorias de las soluciones que se han encontrado. Por tanto, para evitar el ciclado, la lista de corridas se rastreará hacia atrás para determinar todos los movimientos que se considerarán tabú para evitar una solución ya explorada. Para este propósito una secuencia de cancelación residual (RCS) se contruye a pasos mediante una traza. (El rastreo hacia atrás de la lista de corridas se le llamará traza). En cada paso de una traza(paso de traza), exáctamente un atributo se procesa y se empieza con el último(más reciente) hacia el primero.

Se inicia con una RCS vacía, sólo se adicionan aquellos atributos cuyo complemento no está en la sucesión, de otra forma, sus complementos en RCS se eliminan, por lo que en cada paso de traza se conocen qué atributos pueden invertirse para que la solución actual regrese a alguna ya examinada durante el proceso de búsqueda.

Lista de Corridas:1̃ 2̃ 6 4 1 5̃ 2 3 2̃ 5 (último mvto. 5) iteración 10				
Posición	Paso de Traza	RCS	long.	mvto. tabú
10	1	5	1	5̃
9	2	2̃ 5	2	-
8	3	3 2̃ 5	2	-
7	4	3 5	2	-
6	5	3	1	3̃
5	6	1 3	2	-
4	7	4 1 3	3	-
3	8	6 4 1 3	4	-
2	9	2̃ 6 4 1 3	5	-
1	10	2̃ 6 4 3	4	-

Tabla 4.3.1 Sucesión de Cancelación Residual (RCS).

Si los atributos restantes en la RCS pueden invertirse mediante exáctamnete un movimiento, entonces ese movimiento es tabú en la siguiente iteración, puesto que de otra forma este movimiento producirá una solución anterior. La Fig. 4.3.1 proporciona un ejemplo para construir una sucesión de cancelación residual.

Como se había comentado, la asignación del estatus tabú mediante REM es un criterio necesario y suficiente para prevenir revisitar soluciones conocidas.

Suponga que en la iteración i del procedimiento de búsqueda el k-ésimo paso de traza proporciona una RCS de longitud l. Entonces, en la iteración $i+1$ la longitud de la RCS en el paso de traza $k+1$ es igual a $l-1$ o a $l+1$. Por tanto, siguiendo a Glover y a Dammeyer y Voβ, un vector *último* se puede definir como:

$$último(j)=pos\_n,$$

donde *pos_n* es la posición más pequeña tal que el rastreo hasta *pos_n* permite una RCS de longitud $l \leq j+1$.

Si *útimo*(1),...,*último*(p) se han determinado en la iteración l(p $\leq$ i), en la iteración $i+r$ ($r \leq p$) el rastreo se necesita llevar únicamente hasta el número de posición almacenado en *último*(r). Sin embargo, la actualización de *último*(r)=1 debe realizarse en cualquier iteración $i+r$.

Como el número de atributos almacenados se incrementa, puede llegar a ser muy oneroso el trazar la lista de corridas completamente para un número de iteraciónes grande,

por lo que un parámetro *en_n* se introduce para limitar el número de pasos de traza por iteración.

Es claro que no es necesario el trazar la lista de corridas si un atributo del movimiento actual no tiene su complemento en la lista de corridas. Dammeyer y VOβ proporcionan el siguiente criterio de paro del rastreo:

Si en adición a un atributo *solitario* e, un segundo atributo *solitario* $e' \neq e$ se encuentra.

Donde un atributo *solitario* e es aquel que no tiene su complemento $\tilde{e}$ en un cierto número de paso, dado por *último* o por el parámetro *en_n*.

4.3.2 ADAPTACION PARA PROBLEMAS ENTEROS DE VARIABLE ACOTADA

Siguiendo a Glover (1990a), se puede observar que los atributos de movimiento que satisfacen las condiciones necesarias y suficientes descritas anteriormente, resultan un medio natural para crear un atributo para representar el incremento o decremento de cada variable entera para un valor específico. Pero esto produce $2U(j)$ atributos para cada variable entera x_j, donde $U(j)$ es la cota superior de la j-ésima variable, por lo que se puede requerir de arreglos de dimensión muy grande.

Glover propone una estructura de datos alternativa que sólo requiere de un espacio para los arreglos de $3m$, donde m es el número total de variables.

Lo anterior se basa en el hecho de que no es la sucesión de los elementos en la C-sucesión lo importante, sino su identificación. Entonces los arreglos *predecesor* y *sucesor* sirven sólo como un medio conveniente para poder trazar esta identidad.

Para trazar la información en el caso general de cota superior, se utiliza un arreglo *desviación* el cual identifica al vector diferencia $x(i) - x(q)$, *i.e.*, *desviación(j)* $= x_j(i) - x_j(q)$. Se inicia con todas las entradas del arreglo *desviación* iguales a cero. Cada $e(h)$ encontrado durante la traza de $e(q)$ a $e(i)$ en la lista tabú registra la información de que una x_j particular fue incrementada o decrementada, y sobre la base de esta información el correspondiente valor de *desviación(j)* se cambia en dirección opuesta. Sea n el valor de la suma de los valores absolutos de las entradas del arreglo *desviación*. Entonces cuando n=1, el único elemento unido por los arreglos *predecesor* y *sucesor* es el de índice j tal

que *desviación(j)*= 1 ó -1, indicando que la x_j apropiada es tabú para incrementarse o decrementarse respectivamente.

4.4 RESULTADOS COMPUTACIONALES

Cabe señalar que no existe adaptación reportada hasta el momento de la técnica de la búsqueda tabú para el problema de inventarios multiproducto con demanda determinística y costos de reaprovisionamiento conjuntos. Esta adaptación probó su robustes al encontrar mejores soluciones respecto a las reportadas por algoritmos eficientes de la literatura, en particular se compararón los siguientes:

KyR : Kaspi y Rosenblatt (1985), **N-R** : Narro Ramírez (1993).

GyB : Goyal y Belton (1979), **TB** : Búsqueda Tabú.

Las Tablas 4.1 y 4.2 son ejemplos de Narro Ramírez donde se muestra la comparación de los algoritmos.

j	d_j	h_j	k_j	GyB (β_j)	KyR (β_j)	N-R (α_j, β_j)		BT (β_j)
1	3000	2	5	1	1	0.065	1	1
2	20000	2	7	1	1	0.0	1	1
3	12500	1.6	4	1	1	0.506	1	1
4	10000	1.45	5	1	1	0.0	1	1
5	4250	2	5	1	1	0.14	1	1
6	2500	3	7	1	1	0.156	1	1
7	5000	1	10	1	2	0.0	2	2
8	3000	1.25	6	1	1	0.089	1	2
9	2000	3	6	1	1	0.042	1	1
10	1500	2	9	2	2	0.0	2	2
t_1				0.0419	0.04	0.026		0.038
CT				4914.5	4895	4895		4892.85

Tabla 4.4.1 Ejemplo de 10 artículos.

j	d_j	h_j	k_j	KyR (β_j)	N-R (α_j, β_j)		BT (β_j)
1	10000	10	10	1	0.33534	1	1
2	35000	2	8	1	0.23943	1	1
3	8000	8	8	1	0.178644	1	1
4	9000	5	6	1	0.11059	1	1
5	50000	1	8	1	0.0	1	1
6	15000	2	5	1	0.05467	1	1
7	20000	2	7	1	0.6532	1	1
8	12500	1.6	4.5	1	0.0099	1	1
9	10000	1.45	8	1	0.0	1	2
10	4250	2	5	2	0.0	1	2
11	2500	3	7	2	0.0	1	2
12	10000	1	10	1	0.0	1	2
13	4000	1.25	5	1	0.0	1	2
14	2000	3	6	1	0.0	1	2
15	1500	2	9	3	0.0	2	4
CT				11632.35	11772.5		11449.76

Tabla 4.4.2 Ejemplo de 15 artículos.

Para probar la robustes de la eficiencia del algoritmo propuesto en este trabajo se corrieron para cada instancia los tres algoritmos. Se generaron 100 instancias aleatorias para cada uno de los tamaños n = 5, 10, 15, 20, 25 y 30, con los siguientes rangos de las variables:

$$2000 \leq d_j \leq 50000, \; d_j \text{ entero}, \; j = 1, 2, \ldots, n,$$

$$1 \leq h_j \leq 8, \; j = 1, 2, \ldots, n,$$

$$1 \leq k_j \leq 12, \; j = 1, 2, \ldots, n.$$

Para resolver el sistema del Paso 1, se utilizó el paquete GINO, los resultados obtenidos fueron los siguientes:

n		KyR		N-R		BT	
		Costo	Tiempo	Costo	Tiempo	Costo	Tiempo
5	min.	4908	0.0297	4995	0.0310	4908	0.0297
	max.	6928	0.0183	6928	0.0183	6928	0.0183
	prom.	5805	0.0237	5835	0.0249	5805	0.0237
10	min.	8743	0.0216	8791	0.0228	8716	0.0213
	max.	10125	0.0174	10176	0.0182	10064	0.0171
	prom.	9632	0.0179	9664	0.0180	9602	0.0177
15	min.	7992	0.0201	7992	0.0210	7896	0.0207
	max.	11993	0.0144	12059	0.0148	11993	0.0144
	prom.	10448	0.0158	10545	0.0166	10446	0.0160
20	min.	12907	0.0134	13196	0.0145	12864	0.0129
	max.	15767	0.0100	16244	0.0110	15732	0.0099
	prom.	14650	0.0122	15080	0.0134	14609	0.0118
25	min.	13497	0.0188	13709	0.0193	13346	0.0172
	max.	17361	0.0131	17924	0.0144	17309	0.0127
	prom.	15035	0.0166	15345	0.0175	14958	0.0158
30	min.	18593	0.0137	19250	0.0148	18354	0.0123
	max.	21038	0.0128	21769	0.0138	20679	0.0106
	prom.	19947	0.0132	20652	0.0142	19615	0.0111

Tabla 4.4.3 Valores generados por 100 instancias aleatorias.

A continuación se presenta la eficiencia que se calculó tomando como base el valor z_{tabu}, esto es, el valor de la función objetivo obtenido por el algoritmo propuesto en este trabajo y se utilizó la ecuación:

$$eficiencia = 1 - \frac{z - z_{tabu}}{z_{tabu}}.$$

Los resultados son los siguientes:

n		KyR	N-R	BT
5	min.	1	0.982	1
	max.	1	1	1
	prom.	1	0.994	1
10	min.	0.996	0.991	1
	max.	0.993	0.988	1
	prom.	0.996	0.993	1
15	min.	0.999	0.999	1
	max.	1	0.994	1
	prom.	0.999	0.990	1
20	min.	0.996	0.974	1
	max.	0.997	0.967	1
	prom.	0.997	0.967	1
25	min.	0.988	0.972	1
	max.	0.996	0.964	1
	prom.	0.994	0.974	1
30	min.	0.986	0.951	1
	max.	0.982	0.947	1
	prom.	0.983	0.947	1

Tabla 4.4.2 Eficiencia Comparativa.

Se puede observar de esta última tabla que la eficiencia del algoritmo propuesto utilizando un marco de búsqueda tabú es la más elevada. El número de iteraciones bajo el marco de búsqueda tabú que se realizarón fueron de $2n$ donde n es el número de artículos considerados. Cabe señalar que en el estudio de Kaspi y Rosenblatt compararon la efectividad de varios algoritmos generando 132000 ejemplos y cubriendo 132 clases diferentes.

CONCLUSIONES

En este trabajo se partió del estado del arte de la optimización combinatoria, se revisaron los diferentes métodos que en la actualidad se utilizan y se enfocó la atención al análisis de la técnica de la Búsqueda Tabú, observando que esta técnica es aplicable a todo tipo de problemas de programación combinatoria.

Se revisó el problema de calendarización con costos de penalización por retraso y costos de actualización, presentándolo de manera original, como marco para introducir los diferentes elementos de la técnica de la búsqueda tabú. Un aspecto de interés fue el de la identificación de valles profundos, concepto de suma importancia para la eliminación del ciclado y/o soluciones suboptimales.

Se describen tres adaptaciones de la búsqueda tabú para el problema de asignación cuadrático, para el cual no se ha reportado su utilización en la literatura, encontrándose que las propuestas en este trabajo proporcionan mayor eficiencia. En todos los casos se encontrarón en al menos una instancia, los valores óptimos y/o las mejores soluciones reportadas en la literatura, y a diferencia de otros trabajos se reportan el número de iteraciones con puntos iniciales aleatorios y sin utilizar ningún otro procedimiento, para encontrar soluciones iniciales y/o de mejoramiento, reportándose la experiencia computacional.

Aunque nuestro trabajo se basa en un número limitado de estrategias de búsqueda de vecindades así también como de soluciones iniciales, es interesante observar como el uso de criterios de niveles de aspiración así como, el manejo dinámico de las listas tabú proporcionan métodos más robustos de la búsqueda tabú en cuanto a la eficiencia de soluciones para problemas clásicos considerados como difíciles.

Se desarrolla para el caso de los modelos de inventarios multiproducto con demanda determinística y con costos de reaprovisionamiento conjunto, un algoritmo original, no sólo porque es la primera vez que se reporta un esquema de la búsqueda tabú para este tipo de problemas, sino que además proporciona la mayor eficiencia de soluciones con respecto a soluciones proporcionadas por algoritmos reportados en la literatura especializada como de lo más eficiente. Se reporta la eficiencia computacional, observándose que la eficiencia del algoritmo propuesto tiene la mayor para todas las instancias con respecto de los otros algoritmos considerados que a lo sumo llegan a empatar nuestras soluciones.

Las adaptaciones del método de búsqueda tabú son muy variadas como se ha podido observar, y aún con las estructuras más sencillas del mismo se pueden obtener "buenas soluciones" como se puede observar fácilmente en el último capítulo.

A lo largo del trabajo también se ha observado que la técnica de la búsqueda tabú proporciona excelentes resultados para problemas de tipo combinatorio y que existen por explorar muchas otras adaptaciones para obtener cada vez mayor eficiencia.

REFERENCIAS

Barnes J. W. and Vanston L. K. (1981), *Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Setup Costs*, ORSA, 29:1, pp. 146-159.

Barnes J. W. and Laguna M. (1993), *A Tabu Search Experience in Production Scheduling.* Annals of Ops. Res., **41.** pp. 141-156.

Bovet J., Constantin C. and de Werra D. (1991), *A Convoy Scheduling Problem*, Discrete Applied Mathematics, **30**, pp. 1-14.

Brandeau M. l. and Chiu S. S. (1989), *An Overview of representative Problems in Location Research*, Management Science, **35**:6, pp. 645-674.

Bukard R.E. (1990), *Locations with Spatial Interactions: The Quadratic Assigment Problem*, Mirchandani P. B. and Francis R. L.(Eds.), John Willey Sons, pp. 387-438.

Cerny V. (1985), *Thermodynamical Approach to the Traveling Salesman Problem: An efficient simulation algorithm*, Journal of Optimization Theory and Applications, **45**, pp. 41-52.

Chakrapani J. and Skori-Kapov J. (1993), *Massively Parallel Tabu Search for the Quadratic Assignment Problem*, Annals of Ops. Res., **41**, pp. 327-341.

Committee on the Next Decade of Operations Research (Condor 1988), *Operations Reseach: The Next Decade.* Ops. Res., **36** pp. 1-15.

Dammeyer F. and Voβ S. (1993), *Dynamic Tabu List Management Using the Reverse Elimination Method*, Annal of Ops. Res., **41**, pp. 31-46.

de los Cobos Silva S. (1987), *Un Sistema para la Resolución del Problema de Localización de Facilidades*, Tesis de Maestría, C.E.C.-Colegio de Postgraduados, Chapingo México.

de los Cobos Silva S., Venegas Martínez F. y González Santoyo F. (1993), *La Optimización de los Sistemas Productivos: Un Enfoque Globalizador*, Ier. Simposium Internacional de Ingeniería Industrial y de Sistemas, ITESM., Toluca, México.

de los Cobos Silva S. (1994), *La Técnica de la Búsqueda Tabú y sus Aplicaciones*, VII Congreso Latino Ibero Americano de Investigación de Operaciones e Ingeniería de Sistemas, Santiago de Chile.

Friden C., Hertz A. and de Werra D. (1989), *STABULUS: A Technique for Finding Stable Sets in Large Graphs with Tabu Search*, Computing, **42**, pp. 35-44.

Fox B. (1993), *Integrating and Accelerating Tabu Search, Simmulated Annealing, and Genetic Algorithms*,

Annals of Ops. Res., **41**, pp. 47-67.

Glazebrook K. D. and Gittins J.C. (1981), *On Single-Machine Scheduling with Precedence Relatios and Linear or Discounted Cost*, ORSA, 29:1, pp. 161-173.

Glover F. (1965), *A Multiphase-Dual Algorithm for Zero-One Integer Programming Problem*. Ops. Res., **16**, pp. 741-749

Glover F. (1977), *Heuristics for Integer Programming using Surrogate Constrains*, Decision Sciences, **8**, pp. 156-166

Glover F. (1986), *Future Paths for Integer Programming and Links to Artificial Intelligence*, Computers and Operations Research, **13**, pp. 533-549

Glover F. (1987), *Tabu Search*, Preliminary draft: Paper presented at the ORSA/TIMS Joint National Meeting, St. Louis. Center For Applied Artificial Intelligence, University of Colorado

Glover F. (1989), *Tabu Search, Part I*, ORSA Journal on Computing, **1**:3, pp. 190-206.

Glover F. and Greenberg H. J. (1989), *New Approaches for Heuristic Search: A Bilateral Linkage with Artificial Intelligence*, European J. of Opl. Res., **39**, pp. 119-130.

Glover F. (1990), *Tabu Search: A Tutorial*, Interfaces, 20:4, pp. 74-94.

Glover F., Klingman D. and Phillips N. (1990), *Netform Modeling and Applications*, Interfaces, **20**:4, pp. 7-27.

Glover F. (1990a), *Tabu Search, Part II*, ORSA Journal on Computing, **2**:1, pp. 4-31.

Glover F. (1993), *Genetic Algorithms and Scatter Search: Unsuspected Potencials*, Technical report (August), School of Busines, University of Colorado at Boulder.

Glover F. and Laguna M. (1993), *Tabu Search*, Modern Heuristic Techniques for Combinatorial Problems, Colin R. Reeves(Ed.), Blackwell Scientific Publications, Oxford, pp. 70-150.

Glover F., Tailard E. and de Werra D. (1993), *A User's Guide to Tabu Search*, Annals of Ops. Res., **41**, pp. 3-28.

Goyal S.K. and Belton A.S. (1979), *On a Simple Method of Determining Order Qualities under Deterministic Demand*, Management Science, **25**, pp. 604-612.

Gutierrez-Andrade M. (1991), *La técnica del Recocido Simulado y sus Aplicaciones*, Tesis de Doctorado, DEPFI-UNAM., México.

Hadj-Alouane A., Bean J. C. and Murty K. G. (1993), *A Hybrid Genetic/Optimization Algorithm for Task Allocation Problem*, Technical report (November), Dept. of Ind. and Ops. Engineering, The University of Michigan at Ann Arbor.

Hadj-Alouane A. and Bean J. C. (1993), *A Genetic Algorithm for the Multiple-Choice Integer Program*, Technical report (July), Dept. of Ind. and Ops. Engineering, The University of Michigan at Ann Arbor.

Hansen P. (1986), *The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming*, presentado en el Congress on Numerical Methods in Combinatorial Optimization, Capri, Italia.

Hertz A. (1991), *Tabu Search for Large Scale Timetabling Problems*, European J. of Opl. Res., **54**, pp. 39-47.

Hertz A. and de Werra D. (1987), *Using Tabu Search Techniques for Graph Coloring*, Computing, **29**, pp. 345-351.

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

Hooker J.N. (1994), *Needed: An Empirical Science Of Algorithms*, Operation Research, **42**: 2, pp. 201-212.

Kaspi M. and Rodenblatt M.J. (1985), *The Effectiveness of Heuristic Algorithm for Multi-Item Inventory Systems with Joint Replenishment Costs*, International Journal of Production Research, **23**, pp. 109-116.

Kelly J., Golden B. and Assad A. (1993), *Large-Scale Controlled Rounding Using Tabu Search with Strategic Oscillation*, Annals Ops. Res., **41**, pp. 69-84.

Kirkpatrick S., Gelatt C. and Vecchi M. (1983), *Optimization by Simulated Anneling*, Science, **220**, pp. 671-680.

Knuth D. E. (1973), *The Art of Computer Programming*, Addison-Wesley.

Laguna M., Barnes J. W. and Glover F. (1990), *Tabu Search for a Single Machine Scheduling Problem*, Technical report (july), Advanced Knowledge Systems Group of U S, West Advanced Technologies.

Laguna M. (1993), *A Guide to Implementing Tabu Search*, Technical report (October), Graduate School of Business and Administration, University of Colorado, Boulder.

Laguna M. and Kelly J. (1993), *Master Production Scheduling in a Single Facility with Sequence-Dependent Changeover Times*, Technical report (September), Graduate School of Business and Administration, University of Colorado at Bould

Laguna M. and Glover F. (1993), *Integrating Target Analysis and Tabu Search for Improved Scheduling Systems*, Expert Systems with Aplication, **Vol. 6**, pp. 287-297.

Love R.F., Morris J.G. and Wesolowski G.O. (1988), *Facilities Location*, North-Holland.

McCulloch W. and Pitts W. (1943), *A Logical Calculus of the Ideas Immanent in Nervous Activity*, Bulletin of Mathematical Biophysics, **5**, pp. 115-133.

Murty K. G. (1992), *Network Programming*, Prentice Hall.

Narro Ramírez A.E. (1993), *Sistemas de Inventarios Multiproducto.*, reporte técnico(noviembre) DEPFI-UNAM., México.

Nilson N.J. (1987), *Principios de Inteligencia Artificial*, Ed Diaz de Santos S.A.

Nugent C., Vollman T. and Ruml J. (1968), *An Experimental Comparison of Techniques for Assigment of Facilities to Lacations*, Operation Research, **16**, pp. 150-173.

Pearl J. (1985), *Heuristics*, Adisson-Wesley.

Rechenberg I. (1964), *Cybernetic Solution Path of an Experimental Problem*, Royal Aircraft Establishment, Library Translation 1122, Farnborough.

Rechenberg I. (1989), *Artificial Evolution Artificial Intelligence*, Evolution in Medicine; Learning; Principles and Techniques, Chapman Hall.

Reeves C. (1985), *An Improved Heuristic for the Quadratic Assignment Problem*, J. Opl. Res. Soc., **36**:2, pp. 163-167.

Rosenblatt F. (1962), *Principles of Neurodynamics*, Spartan, Washington, DC.

Salkin H.M. and Mathur K. (1989), *Foundations of Integer Programming*, North-Holland.

Salomon M., Kuik R. and Wassenhove L. (1993), *Statistical Search Methods for Lotsizing Problems*, Annals of Ops. Res., **41**, pp. 453-448.

Skorin-Kapov J. (1990), *Tabu Search Applied to the Quadratic Assignment Problem*, ORSA J. on Computing, **2**:1, pp. 33-40.

Widmer M. and Hertz A. (1989), *A new Heuristic Method for the Flow Shop Sequencing Problem*, European J. of Opl. Res., **41**, pp. 186-193.

Woodruff D. and Zemel E. (1993), *Hashing Vectors for Tabu Search*, Annals of Ops. Res., **41**, pp. 123-137.