Universidad de Los Andes
Facultad de Ingeniería
Doctorado en Ciencias Aplicadas
Seminario Doctoral

Métodos heurísticos para la planificación y el manejo forestal

Por: María Alejandra Quintero M.
Profesores: Dr. Mauricio Jerez
Dra. Magdiel Ablan

Mérida, noviembre de 2009

ÍNDICE

INTRODUCCIÓN

Cuando se desea resolver un problema de optimización existen dos maneras básicas de abordarlo, utilizando un método exacto o mediante la aplicación de métodos heurísticos. Los métodos exactos proporcionan una solución óptima del problema y utilizan algoritmos contrastados y de propósito general, tales como el método simplex para resolver programación lineal, o el algoritmo de ramificación y acotamiento utilizado en programación entera. Sin embargo, muchos de los problemas que se presentan en la práctica no pueden ser resueltos mediante métodos exactos; debido a su complejidad, el tiempo requerido para hacerlo es extremadamente alto, aumentando exponencialmente con el tamaño del problema. A estos problemas se les conoce como "difíciles de resolver" o en lenguaje matemático "NP-hard".

En contraposición a los métodos exactos, los métodos heurísticos se limitan a proporcionar una buena solución del problema, no necesariamente óptima, pero en un tiempo de computación aceptable. En problemas difíciles de resolver, han demostrado ser más eficaces y encuentran buenas soluciones en forma mucho más rápida.

El destacable éxito de los métodos heurísticos para resolver problemas de optimización de difícil solución, especialmente aquellos que surgen en aplicaciones del mundo real, ha causado una explosión de nuevas aplicaciones durante los últimos años, entre las cuales se encuentran algunas experiencias en el sector forestal.

Las técnicas heurísticas se han utilizado en planificación forestal debido al tamaño y complejidad de muchos problemas. Son útiles para desarrollar planes forestales que comprenden objetivos espaciales, en modelos de gran tamaño, y en aquellos problemas que tienen relaciones complejas o no lineales que deben considerarse simultáneamente en la programación de actividades. En la literatura forestal pueden encontrarse diferentes aplicaciones de los métodos heurísticos, de allí la importancia de conocer los aspectos más importantes de este enfoque de solución.

En el capítulo I de esta monografía se abordan los conceptos básicos relacionados a los métodos heurísticos: heurística y metaheurística, situaciones en las que puede ser conveniente utilizar heurísticas, inconvenientes de estas técnicas, tipos de metaheurísticas, y

principios de la búsqueda por entorno, los cuales son claves para comprender el funcionamiento de heurísticas más avanzadas. Igualmente, en este capítulo se incluye una revisión sobre el uso de heurísticas en la planificación forestal, beneficios y aplicaciones de estas técnicas en los diferentes niveles de planificación.

Los siguientes capítulos comprenden una exposición de los aspectos fundamentales de tres de las metaheurísticas clásicas que han tenido mayor aplicación en la planificación y manejo forestal: recocido simulado o *simulated annealing* en inglés (capítulo II), búsqueda tabú (Capítulo III) y algoritmos genéticos (Capítulo IV).

En cada metaheurísica se hace referencia al origen y principios de la técnica, se describe el algoritmo básico, se definen los parámetros, se explican las estrategias o decisiones que deben tomarse para implementar una aplicación del algoritmo, y se hace una revisión bibliográfica acerca de sus aplicaciones en la planificación y manejo forestal.

En la última sección del capítulo II se plantea un ejemplo de aplicación, en el cual se desea conseguir un plan de aprovechamiento de costo mínimo para una unidad forestal conformada por un conjunto de rodales, de tal manera que satisfaga la demanda anual de madera requerida por una planta de procesamiento de pulpa. Se presentan los datos del problema y se utiliza el algoritmo de recocido simulado para resolverlo.

El mismo problema se utiliza también como ejemplo de aplicación en los capítulos dedicados a la búsqueda tabú y a los algoritmos genéticos. El objetivo principal es ilustrar a través de un problema de planificación forestal sencillo cómo llevar a la práctica los conceptos estudiados. Además, con los resultados obtenidos es posible hacer algunas comparaciones entre los diferentes métodos de solución empleados.

Al final de la monografía, se presentan algunas conclusiones que dejan claros la importancia y el potencial de las técnicas heurísticas y en especial del recocido simulado, la búsqueda tabú y los algoritmos genéticos, como métodos alternativos de solución a problemas de optimización que son difíciles o imposibles de resolver mediante los algoritmos exactos, siempre haciendo énfasis en el campo de la planificación forestal.

CAPÍTULO I

Conceptos básicos sobre métodos heurísticos y su utilidad en planificación forestal

1.1 Definición de heurística

La idea más general del término *heurística* está relacionada con la tarea de resolver inteligentemente problemas reales usando el conocimiento disponible. La palabra heurística proviene del griego *Heurískein* que puede traducirse por encontrar, descubrir, hallar (Melián et al, 2003).

Una definición intuitiva es la propuesta por Zeneckis et al. (1981):

> *"Las heurísticas son procedimientos simples basados en el sentido común que se supone que obtendrán una buena solución (no necesariamente la mejor) a un problema difícil de un modo sencillo y rápido".*

En Investigación de Operaciones, una heurística es una técnica de resolución de problemas de optimización (conformada por una regla o un conjunto de reglas) que busca buenas soluciones a un costo computacional razonable, sin garantizar optimalidad. Una característica de los métodos heurísticos es que pueden encontrar una solución de alta calidad pero que no necesariamente es la óptima, e incluso, en muchos casos no se llega a establecer lo cerca que está una solución heurística de la solución óptima.

En una primera aproximación puede parecer que existen serios inconvenientes relacionados al uso de los métodos heurísticos, sin embargo, es necesario enfatizar que muchas técnicas heurísticas brindan soluciones muy buenas en la práctica.

Cabe destacar el interés creciente por el estudio y aplicación de procedimientos heurísticos en Investigación de Operaciones, pasando de ser considerados pobres herramientas a instrumentos fundamentales y, en muchos casos, imprescindibles para la resolución práctica de un problema.

1.2 Cuándo utilizar heurísticas

El uso de un procedimiento heurístico en lugar de un procedimiento exacto para resolver un problema está especialmente indicado cuando:

1. El problema es de una naturaleza tal que no se conoce algún método exacto para su resolución.

2. Aunque existe un método exacto para resolver el problema, su uso es computacionalmente muy costoso.

3. Se prefiere abordar, por medio de heurísticas, una representación más ajustada del problema planteado que una versión menos realista de tal problema que pueda resolverse de forma exacta. Es decir, se prefiere resolver de forma aproximada un modelo ajustado a la realidad que resolver de forma exacta un modelo aproximado de la realidad. En este sentido, los métodos heurísticos son más flexibles que los métodos exactos, permitiendo la incorporación de condiciones de difícil modelización.

4. Se desea aumentar la eficiencia de un procedimiento exacto. En ese caso un método heurístico puede proporcionar una buena solución inicial de partida o participa en un paso intermedio del procedimiento, como por ejemplo, las reglas de selección de variables de entrada en el método simplex, o el establecimiento de cotas en el algoritmo de ramificación y acotamiento.

5. Se tiene que resolver repetidas veces un mismo problema, probablemente con datos distintos. En estas circunstancias se requiere un procedimiento eficiente de resolución que suministre de forma rápida una solución suficientemente buena. Estas situaciones se presentan, por ejemplo, al tener que determinar diariamente valores de las variables de interés.

1.3 Inconvenientes de las técnicas heurísticas

A pesar de que las heurísticas son una alternativa excelente para resolver problemas de difícil solución, éstas también adolecen de ciertas propiedades deseables. Un inconveniente de la gran mayoría de los métodos heurísticos es su dependencia de la estructura del problema para el cual fue diseñado, y su falta de habilidad para adaptarse a nuevas situaciones o modificaciones del problema de partida. Así, usan propiedades de la región

factible y/o de la función objetivo o información a priori que hacen que los procedimientos sean válidos sólo bajo esas condiciones (Moreno, 1999).

A diferencia de los métodos exactos, no existe un procedimiento conciso y preestablecido, independiente del problema. En los métodos heurísticos las técnicas e ideas aplicadas a la resolución de un problema son específicas de éste y aunque, en general, pueden ser trasladadas a otros problemas, han de particularizarse en cada caso.

Otro problema importante de los heurísticos es su incapacidad para escapar de óptimos locales. Una solución heurística para un problema podría corresponder o estar muy cercana a un óptimo local y no a un óptimo global, ya que estos algoritmos pueden quedarse atrapados en una zona del espacio de soluciones.

Para superar los inconvenientes que presentan los métodos heurísticos clásicos, durante los últimos años diferentes investigadores han estado trabajando en un tipo de procedimientos heurísticos generales que pueden usarse para resolver una amplia variedad de problemas adaptando convenientemente los elementos que los definen y que permiten obtener mejores resultados. Tales procedimientos son las *Metaheurísticas.*

1.4 Metaheurísticas

Las metaheurísticas son estrategias inteligentes para diseñar o mejorar procedimientos heurísticos generales con un alto rendimiento. El término fue introducido por Glover (1986) en el artículo que presenta las ideas básicas de búsqueda tabú. A partir de entonces, han surgido multitud de propuestas para diseñar buenos procedimientos para resolver ciertos problemas que, al ampliar su campo de aplicación, han adoptado la denominación de metaheurísticas.

En la literatura se pueden encontrar diferentes definiciones de meteheurísticas, se cita la propuesta por Osman y Kelly (1996) por ser bastante ilustrativa:

> *"Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los Metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes*

conceptos derivados de los heurísticos clásicos, la inteligencia artificial, la evolución biológica, sistemas neuronales y mecánica estadística".

Otra definición interesante es la de Voß (2001), ya que trata de resumir los aspectos más importantes de las metaheurísticas que han señalado otros autores en sus definiciones:

> *"Una metaheurística es un proceso iterativo que dirige y modifica las operaciones de otras heurísticas subordinadas para producir soluciones de alta calidad. Puede manipular una solución única (completa o incompleta) o un conjunto de ellas en cada iteración. El heurístico subordinado puede ser un procedimiento de alto (bajo) nivel, una simple búsqueda local o un método de construcción"*.

De acuerdo a esta definición, las metaheurísticas se sitúan conceptualmente por encima de las heurísticas en el sentido que guían el diseño de éstas. Así, al abordar un problema de optimización, se puede escoger cualquier metaheurística para diseñar un algoritmo específico que lo resuelva aproximadamente (Martí, 2003).

Las metaheurísticas han tenido un gran desarrollo y crecimiento desde sus comienzos en los años 80. En la práctica, han tenido éxito en una gran variedad de problemas de optimización combinatoria difíciles, siendo en muchos casos la única alternativa factible para encontrar una solución de calidad en un tiempo razonable. En general, las metaheurísticas se comportan como métodos robustos y eficientes, por lo que hoy en día constituyen un área importante de investigación y aplicación.

La familia de las metaheurísticas incluye pero no está limitada a: procedimientos de memoria adaptativa; búsqueda tabú; métodos GRASP (*greedy random adaptive search*); recocido simulado (*simulated annealing*); sistemas de hormigas; búsqueda por entornos variables (*VNS*); métodos evolutivos entre los cuales se encuentran algoritmos genéticos, estrategias evolutivas, búsqueda dispersa (*scatter search*) y algortimos meméticos; redes neuronales; método de aceptación por umbrales (*threshold accepting*); y métodos híbridos.

1.5 Clasificación de las Metaheurísticas

En la literatura se pueden encontrar muchas clasificaciones de las metaheurísticas. La clasificación que aquí se presenta es la propuesta por Melián et al. (2003).

Las metaheurísticas pueden clasificarse de acuerdo a los procedimientos que utilizan en: metaheurísticas de relajación, metaheurísticas constructivas, metaheurísticas de búsqueda y metaheurísticas evolutivas.

1) **Metaheurísticas de relajación:** utilizan relajaciones del modelo original que hacen al problema más fácil de resolver. Una relajación es un modelo simplificado obtenido al eliminar, debilitar o modificar restricciones u objetivos del problema real. Los modelos muy ajustados a la realidad suelen ser muy difíciles de resolver, y sus soluciones difíciles de implementar exactamente, por lo que se acude a modelos relajados. En ese sentido, las metaheurísticas de relajación son estrategias para el empleo de relajaciones y procedimientos para proponer soluciones heurísticas del problema original a partir del modelo relajado. Dentro de este tipo de metaheurísticas se encuentran los métodos de relajación lagrangiana o de restricciones subordinadas.

2) **Metaheurísticas constructivas:** tratan de obtener una solución a partir del análisis y selección paulatina de las componentes que la forman. Constan de un procedimiento que incorpora iterativamente elementos a una estructura, inicialmente vacía, que representa la solución. Las metaheurísticas constructivas establecen estrategias para seleccionar las componentes con las que se construye una buena solución del problema. Entre estas se encuentran la estrategia voraz (greedy) y los métodos GRASP en su primera fase.

3) **Metaheurísticas de búsqueda:** establecen estrategias para recorrer el espacio de soluciones del problema transformando de forma iterativa una solución de partida. Este el tipo de metaheurística más importante y abarca una gran variedad de métodos, entre los que están los métodos de búsqueda local y los de búsqueda global. En los métodos de **búsqueda local** se parte de una solución inicial e iterativamente se trata de mejorar la solución hasta que no sea posible obtener mejoras. La mejora de una solución se obtiene en base al análisis de soluciones similares o cercanas, denominadas soluciones vecinas. Tales métodos se conocen también como búsquedas monótonas (descendentes o ascendentes) y algoritmos escaladores (*hill-climbing*). El principal inconveniente de estas búsquedas locales es que se quedan atrapadas en un óptimo local. Por ello, el propósito de las primeras metaheurísticas

era extender una búsqueda local para continuarla más allá de los óptimos locales, denominándose búsqueda global.

Las metaheurísticas de **búsqueda global** incorporan pautas para escapar de los óptimos locales de baja calidad. Estas pautas consideran tres formas básicas: volver a iniciar la búsqueda desde otra solución de arranque (metaheurísticas de arranque múltiple, *multistart*); modificar en forma sistemática la estructura de entornos que se está aplicando, es decir, la manera cono se define una solución vecina (metaheurísticas de entorno variable*; VNS, Variable Neighborhood Search*); y permitir movimientos o transformaciones de la solución de búsqueda que no sean de mejora (*metaheurísticas no monótonas*). En este último grupo se encuentran las metaheurísticas de estrategias probabilísticas, siendo el recocido simulado (*Simulated Annealing*) la más representativa, y las estrategias con memoria, representadas por la búsqueda tabú (*Tabu Search*).

4) **Metaheurísticas evolutivas:** son procedimientos basados en conjuntos de soluciones, llamados poblaciones, que evolucionan en el espacio de búsqueda con el fin de acercarse a una solución óptima utilizando aquellos elementos que sobreviven en las continuas generaciones de poblaciones. La característica fundamental de estas metaheurísticas es la interacción entre los elementos del conjunto de soluciones que evolucionan, la cual consiste en la combinación entre las soluciones que conforman una población para crear una población nueva que permita evolucionar a la población anterior. Existen dos formas fundamentales de combinar esta información para producir nuevos elementos: procedimientos sistemáticos y procedimientos aleatorios. Entre las metaheurísticas que utilizan procedimientos sistemáticos están los algoritmos genéticos, los algoritmos meméticos y los algoritmos de estimación de distribuciones (EDA); mientras que los métodos de reencadenamiento de caminos (*Path-Relinking*) y la búsqueda dispersa (*Scatter Search*) se encuentran entre las metaheurísticas evolutivas que usan procedimientos sistemáticos.

5) **Otros tipos de metaheurísticas:** otras metaheurísticas que aparecen en las clasificaciones corresponden a tipos intermedios entre los anteriores. Entre ellas destacan las metaheurísticas de descomposición y las de memoria a largo plazo. Las metaheurísticas de descomposición establecen pautas para resolver el problema dividiéndolo en subproblemas, a partir de los que se construye una solución del

problema original, son procedimientos intermedios entre las metaheurísticas de relajación y las constructivas. Las metaheurísticas de memoria a largo plazo constituyen el caso más relevante de las metaheurísticas de aprendizaje y se sitúan entre las de arranque múltiple y las derivadas de la búsqueda tabú.

De acuerdo a Melián et al. (2003), todas las metaheurísticas, de una u otra forma, se pueden concebir como estrategias aplicadas a procesos de búsqueda, en las que para resolver el problema se van modificando elementos del espacio de búsqueda a medida que se aplican las distintas operaciones diseñadas para llegar a la solución definitiva. Es frecuente interpretar que el término metaheurística es aplicable esencialmente a los procedimientos de búsqueda sobre un espacio de soluciones alternativas. Por tal razón, y con el fin de facilitar la comprensión de las técnicas que serán tratadas en esta monografía, en el siguiente apartado se describe con más detalle el proceso de búsqueda por entorno, el cual es la base de muchas de las metaheurísticas.

1.6 Búsqueda por entorno

Los métodos de búsqueda por entorno (o búsqueda local) conforman una clase general de heurísticas basadas en el concepto de explorar el vecindario de la solución actual. Estos procedimientos comienzan con una solución inicial a partir de la cual recorren el espacio de soluciones haciendo un conjunto de modificaciones o movimientos. Las soluciones que se obtienen de otra mediante uno de los movimientos posibles se denominan *vecinas* de ésta y constituyen su *entorno*. El conjunto de movimientos posibles da lugar a una relación de vecindad y una *estructura de entornos* en el espacio de soluciones.

Definiciones

En el contexto de la búsqueda por entorno, una solución del problema se puede definir como un vector **X**; el conjunto de todas las soluciones factibles se denota por $\boldsymbol{U}$; y el costo de una solución $\mathbf{X} \in \boldsymbol{U}$ se denota por $f(\mathbf{X})$ –llamada usualmente función objetivo.

Cada solución $\mathbf{X} \in \boldsymbol{U}$ tiene un conjunto de vecinos, $N(\mathbf{X}) \subset \mathbf{U}$, llamado el **entorno** de **X**.

Cada solución $\mathbf{X'} \in N(\mathbf{X})$ puede ser alcanzada directamente de **X** mediante una operación llamada **movimiento**, y se dice que **X** se mueve a **X'** cuando tal operación es ejecutada.

Una **estructura de entornos** es una función N: $\boldsymbol{U} \rightarrow \boldsymbol{U}$ que asocia a cada solución $\mathbf{X} \in \mathbf{U}$ un entorno $N(\mathbf{X}) \subset \mathbf{U}$. La elección de la estructura de entornos es fundamental en el éxito de los procesos de búsqueda ya que determina la calidad de los movimientos aplicados.

Ejemplo: considérese un problema de planificación forestal en el cual se tienen M unidades de manejo (rodales), N alternativas de manejo para cada unidad, y se desea maximizar el valor presente total (VAN) sujeto a restricciones de producción mínimas y máximas, y a restricciones de singularidad (se adopta un único régimen de manejo para cada rodal). Las variables de decisión son binarias y están definidas así:

$$X_{ij} = \begin{cases} 1 & \text{La alternativa de manejo } j \text{ es asignada al rodal } i \\ 0 & \text{En otro caso} \end{cases}$$

$i = 1 \ldots M$; $j = 1 \ldots N$

Para mostrar cómo pueden generarse soluciones vecinas en este problema, supóngase por simplicidad que M = 4 rodales y N= 3 alternativas de manejo. Una solución puede representarse como un vector de las variables de decisión:

X_{11}	X_{12}	X_{13}	X_{21}	X_{22}	X_{23}	X_{31}	X_{32}	X_{33}	X_{41}	X_{42}	X_{43}
rodal 1			rodal 2			rodal 3			rodal 4		

Figura 1.1 Representación de la solución del problema como un vector

En el grupo de variables que están relacionadas a un rodal (tres variables consecutivas con igual índice i), sólo puede haber una variable igual a 1 y las dos restantes deben ser iguales a 0 para satisfacer la restricción de singularidad, es decir, sólo se asigna una alternativa de manejo a un rodal.

Una posible solución **X** se puede representar tal como se indica en la figura 1.2.

X	0	0	1	0	1	0	0	1	0	1	0	0

Figura 1.2 Representación de una solución como un vector de variables binarias.

De acuerdo a la solución que se muestra en la figura 1.2, al rodal 1 se le asigna la alternativa de manejo 3, a los rodales 2 y 3 les corresponde la alternativa de manejo 2 y al rodal 4 se le asigna la alternativa 1.

Para generar una solución vecina de **X** se pueden utilizar diferentes estrategias cuyo diseño depende de la naturaleza del problema y del ingenio del planificador. Una estrategia sencilla para este caso puede ser elegir aleatoriamente un rodal, cambiar la alternativa de manejo que tiene asignada (*j*) y elegir la alternativa siguiente (*j+1*), en otras palabras, en el rodal elegido, a la variable tiene valor 1 asignarle 0, y a la siguiente asignarle 1. Si el rodal seleccionado tiene asignada la alternativa de manejo 3 en la solución actual, en la solución vecina se selecciona la alternativa 1 ya que no existe una cuarta alternativa. Siguiendo esta estrategia, supóngase que se selecciona aleatoriamente el rodal 2, entonces una solución vecina **X'** obtenida a partir de la solución **x** es la que se muestra en la figura 1.3.

Solución actual X	0	0	1	0	1	0	0	1	0	1	0	0
Solución vecina X'	0	0	1	0	0	1	0	1	0	1	0	0

Figura 1.3 Representación de una solución vecina generada mediante el cambio de alternativa de manejo en uno de los rodales.

Otra estrategia puede ser elegir al azar cierta cantidad de rodales y en cada uno de ellos se selecciona aleatoriamente una alternativa distinta a la solución actual. Un ejemplo de la aplicación de esta estrategia se observa en la figura 1.4, en la que se seleccionaron aleatoriamente dos rodales, el 3 y el 4, y en cada uno de ellos se seleccionó al azar una alternativa de manejo distinta a la solución actual.

Solución actual X	0	0	1	0	1	0	0	1	0	1	0	0
Solución vecina X'	0	0	1	0	1	0	1	0	0	0	1	0

Figura 1.4 Representación de una solución vecina generada mediante el cambio de alternativa de manejo en dos rodales.

En general, para cualquier problema específico hay diferentes estructuras de entorno posibles, dependiendo de la estrategia de movimientos utilizada y de la definición del espacio de búsqueda (Gendreau, 2003).

Una vez diseñada la estrategia que se utilizará para generar movimientos que conducen a la obtención de soluciones vecinas se puede aplicar el proceso iterativo de búsqueda.

El esquema general de un procedimiento de búsqueda por entornos consiste en generar una solución inicial y, hasta que se cumpla un criterio de parada, seleccionar iterativamente un movimiento para modificar la solución. Las soluciones son evaluadas mientras se recorren y se propone la mejor solución encontrada. En la figura 1.5 se presenta un diagrama de flujo básico para el proceso de búsqueda por entornos.

Otros aspectos que deben decidirse para aplicar este algoritmo además de la estructura de entornos, son la forma de generar la solución inicial y el criterio de parada.

Algunas metaheurísticas tal como el recocido simulado y la búsqueda tabú, se basan en la inclusión de procedimientos y estrategias que mejoran el proceso de búsqueda por entornos, evitando que se éste se quede atrapado en un óptimo local.

1.7 Uso de heurísticas en planificación forestal

Los modelos de programación matemática se han venido utilizando con éxito en la planificación y manejo forestal desde finales de los 60. Durante la década de los años 80, técnicas típicas de la investigación de operaciones, como la programación lineal y sus variantes (programación entera, programación por metas) comenzaron a reemplazar a los métodos clásicos de planificación forestal. El uso de estas técnicas conjuntamente con técnicas de simulación basadas en modelos de crecimiento mejoró la capacidad de análisis respecto a las múltiples alternativas de decisión existentes en los problemas de planificación forestal (Palahí y Pukkala, 2004).

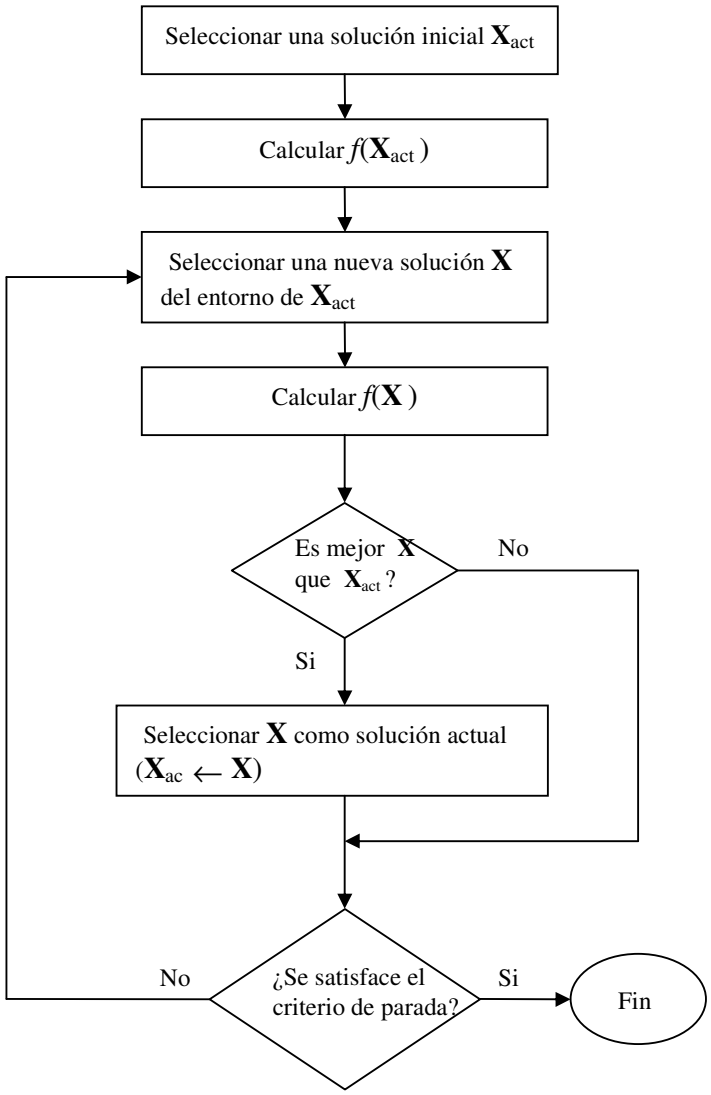


Figura 1.5 Pasos básicos de la búsqueda por entorno

Sin embargo, la programación matemática se ha encontrado con limitaciones para resolver problemas actuales cada vez más complejos, como aquellos que incluyen aspectos espaciales, múltiples objetivos, relaciones funcionales no lineales, o problemas cuyo tamaño los hace muy difíciles de resolver.

Los requerimientos espaciales están relacionados al tamaño, forma y yuxtaposición de las unidades de manejo (Baskent, 2001). Por ejemplo, en algunos problemas de planificación es necesario incluir restricciones que garanticen que un rodal no puede ser cortado hasta que

los rodales adyacentes no hayan alcanzado cierta madurez, o que impidan que dos áreas contiguas sean aprovechadas en el mismo período. Estas restricciones se denominan restricciones de adyacencia y evitan combinaciones no deseadas de las actividades de corta en áreas adyacentes. Las restricciones espaciales complican los problemas de planificación, ya que usualmente es necesario incluir un gran número de ellas, limitando el uso de las técnicas clásicas de programación matemática.

Una tendencia en los problemas actuales de planificación forestal es diseñar planes que traten de optimizar diferentes objetivos: económicos, ecológicos y sociales. En este tipo de problemas se considera que un bosque además de la producción de madera tiene otras funciones, tales como conservación de la biodiversidad, fijación de carbono, protección del suelo, regulación de recursos hídricos, recreación, obtención de productos no madereros como miel, frutos, pastos, entre otros (Palahí et al., 2004). A este enfoque se le denomina planificación forestal multiobjetivo y los modelos resultantes son difíciles de resolver.

En algunos problemas se desea incluir en el proceso de planificación relaciones cuantitativas que no son fácilmente descritas por medio de relaciones lineales. Hay relaciones en los sistemas forestales que tienen una naturaleza no lineal, por lo que no pueden incluirse en los modelos clásicos de programación matemática sin hacer una simplificación (Bettinger et al., 2009). En muchos casos es preferible utilizar métodos alternativos que permitan reflejar de manera más precisa las características del sistema en estudio.

Otra limitación importante de los métodos clásicos de programación matemática está relacionada al tamaño del problema en estudio. Un problema de planificación forestal puede tener un alto grado combinatorio, es decir, el número de posibles soluciones es tan grande que resolverlo mediante un algoritmo exacto, aún utilizando computadoras con gran capacidad de cálculo y software adecuado, puede tomar mucho tiempo sin que se llegue a una solución, además pueden surgir inconvenientes (como desbordamiento de memoria) que hacen que la ejecución se interrumpa. Supóngase un problema en el que se desea saber cuántos rodales de 20 se van a aprovechar en un año determinado, en este caso sencillo se tienen 2^{20} (1049×10^6) posibles soluciones, lo que dificulta la aplicación de los métodos exactos, y si se consideran más períodos de tiempo y algunos cientos de rodales, el problema llega a ser intratable por los métodos exactos.

Las limitaciones de los métodos de programación matemática en la solución de problemas de planificación forestal han propiciado el uso de técnicas heurísticas como alternativa para obtener una buena solución en un tiempo de computación razonable. Tal como lo señalan Palahí y Pukkala (2004):

> *"La planificación forestal debe incluir criterios cada vez más diversos y complejos con el fin de asegurar una gestión ecológicamente respetuosa, socialmente aceptable y económicamente viable. Este nuevo paradigma requiere de un mayor uso de objetivos espaciales y de relaciones no lineales en el diseño y formulación de los problemas de planificación forestal para reproducir la realidad lo más verazmente posible. En este contexto, las técnicas clásicas de optimización dejan muchas veces de ser efectivas y métodos heurísticos son necesarios para obtener soluciones de alta calidad en un tiempo razonable".*

En las últimas dos décadas se han realizado numerosas investigaciones en las que se hace uso de métodos heurísticos para resolver problemas de planificación forestal complejos. Entre los métodos heurísticos que se han aplicado en la planificación forestal se encuentran el método de Monte Carlo, y las metaheurísticas recocido simulado (*simulated annealing*), búsqueda tabú (*tabu search*), y algoritmos genéticos

Las heurísticas se han utilizado en los distintos niveles de la planificación forestal: estratégico, táctico y operacional. En los tres niveles existe una existe una multiplicidad de factores que deben considerarse al momento de tomar decisiones, lo que se traduce en problemas complejos que en mucho casos requieren el uso de técnicas heurísticas para ser resueltos.

La planificación estratégica comprende la programación de actividades para áreas extensas y a largo plazo, es generalmente de naturaleza no espacial y está relacionada a establecimiento de políticas globales de manejo forestal tales como disponibilidad de tierras y recursos, tipos de tratamientos a aplicar, definición de la capacidad de producción. La planificación forestal a nivel estratégico trata de responder la pregunta: ¿cuál debería ser el objetivo general del bosque?. Aunque el enfoque predominante para elaborar planes forestales estratégicos ha sido la programación lineal, en los años 90 se comenzó a utilizar métodos heurísticos, principalmente la metaheurística recocido simulado; pueden citarse los trabajos

de Lockwood y Moore (1993), Van Deusen (1999) y los modelos desarrollados en el estado de Oregon, EEUU, para manejar la mayoría de sus bosques.

La planificación táctica se refiere a decisiones para un período de tiempo más corto y con mayor nivel de detalle, como el desarrollo de planes de aprovechamiento, el diseño y habilitación de caminos, la definición de tratamientos silviculturales, la elaboración planes de distribución de la madera, entre otras. En la planificación táctica las heurísticas han sido las técnicas dominantes, debido en parte a las consideraciones espaciales que usualmente se incluyen en este tipo de planes. Las restricciones de tipo espacial requieren el uso de un gran número de variables enteras, por lo que los métodos heurísticos son la alternativa más adecuada para resolver estos modelos. Los algoritmos heurísticos usados a nivel táctico incluyen el método de Monte Carlo, búsqueda tabú, algoritmos genéticos, recocido simulado y otros métodos de búsqueda por entorno estocásticos, también se han diseñado heurísticas específicas para problemas particulares.

Por su parte, la planificación operacional realiza una especificación más detallada de las alternativas, a este nivel se establece la programación de las actividades que se ejecutarán a corto plazo, abarcando una amplia variedad de decisiones tales como la selección y localización de la maquinaria de cosecha, contratación de mano de obra, asignación de trabajos, diseño de rutas para el transporte de madera, decisiones de corte de trozas, entre otras. La mayoría de las aplicaciones a nivel operacional hace uso de heurísticas, destacándose el recocido simulado, la búsqueda tabú y los algoritmos genéticos.

En la práctica, las técnicas heurísticas han tenido limitaciones de uso debido a la poca disponibilidad de software. Las heurísticas comúnmente son diseñadas para resolver un problema particular, como resultado, el software desarrollado para un caso específico es difícil de utilizar en otro problema, de allí que no existan paquetes de computación estándar que puedan usarse de manera general (Bettinger, et al., 2009). Incluso, cuando se trabaja con metaheurísticas que son técnicas que pueden aplicarse a una amplia variedad de problemas, es complicado usar paquetes de computación desarrollados previamente, ya que cada caso tiene sus particularidades, por ejemplo, la definición del espacio de soluciones y la estructura de entornos puede ser distinta en cada problema.

Los beneficios que aportan las técnicas heurísticas se derivan de la posibilidad de resolver problemas de planificación más realistas y menos rígidos que los que se pueden plantear con la aplicación de técnicas derivadas de la programación matemática.

CAPÍTULO II

Recocido Simulado (*Simulated Annealing*)

2.1 Origen del Recocido Simulado

La técnica heurística del recocido simulado (RS), en inglés *Simulated Anneling*, está basada en un algoritmo propuesto por Metropolis et al. (1953) en el marco de la termodinámica estadística, para simular el proceso de enfriamiento de un material (recocido). Kirkpatrick et al. (1983) e independientemente Cerni (1985) establecieron una analogía entre el proceso de recocido y el reto de resolver problemas de optimización combinatoria de gran escala. Desde entonces el recocido simulado ha sido utilizado para resolver en forma exitosa una amplia variedad de problemas de optimización combinatoria, convirtiéndose en una metaheurística clásica.

2.2 Principios del método

El proceso de recocido es una estrategia para modificar el estado de un material y alcanzar un estado óptimo mediante el control de la temperatura. El recocido comienza con el calentamiento del material a una alta temperatura, para luego enfriarlo lentamente, manteniendo en cada etapa una temperatura por cierto tiempo. Si la disminución de la temperatura es demasiado rápida, pueden originarse defectos en el material. Esta técnica de disminución controlada de la temperatura conduce a un estado sólido cristalizado, el cual es un estado estable y que corresponde a un mínimo absoluto de energía (Dréo et al., 2006).

Metrópolis et al. (1953) modelaron el proceso de recocido simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura, hasta que converge a un estado estable. De acuerdo a las leyes de termodinámica, la probabilidad de un incremento de energía ΔE a una temperatura t se puede aproximar por:

$$P(\Delta E) = e^{-(\Delta E / kt)} \quad [2.1],$$

donde k se denomina constante de Boltzman.

En el algoritmo de Metropolis se genera una modificación en el sistema y se calculan los cambios de energía resultantes; si esta transformación origina una disminución energética,

se acepta la modificación, y si ocurre un incremento de energía, el cambio será aceptado con una probabilidad dada por [2.1]. A una temperatura alta la probabilidad $P(\Delta E)$ es cercana a 1, por lo que la mayoría de los cambios generados en el sistema son aceptados; a una temperatura baja $P(\Delta E)$ es cercana a 0 y la mayoría de los cambios son rechazados ((Dowsland y Díaz, 2001; Dréo et al., 2006).

El uso del recocido simulado en optimización combinatoria se basa en establecer analogías entre el sistema físico y el problema de optimización. En la tabla 2.1 se muestran estas analogías.

Tabla 2.1. Analogía entre el sistema físico y el problema de optimización

Sistema físico (termodinámica)	**Problema de optimización**
Estados del sistema	Soluciones factibles
Energía	Función de costo (función objetivo)
Cambio de estado	Solución vecina
Temperatura	Parámetro de control T
Estado estable	Solución óptima

El método recocido simulado transpone el proceso de recocido a la solución de un problema de optimización, la función objetivo del problema, similar a la energía del material, es minimizada con la ayuda de una temperatura ficticia, la cual es un parámetro de control del algoritmo. Este parámetro debe tener el mismo efecto que la temperatura del sistema físico: conducir hacia el estado óptimo. Si la temperatura es disminuida gradualmente y de manera controlada se puede alcanzar el mínimo global, si es disminuida abruptamente se puede llegar a un mínimo local.

2.3 Algoritmo de recocido simulado

El algoritmo de recocido simulado parte de una solución factible $\mathbf{X}_{act}$ que puede ser generada en forma aleatoria y se evalúa la función de costo (función objetivo) para esa solución, llámese a este valor $f(\mathbf{X}_{act})$. Se comienza con una temperatura ficticia alta y se genera una nueva solución $\mathbf{X}$ que corresponde a una solución vecina de $\mathbf{X}_{act}$, para la cual se calcula el valor asociado de la función de costo $f(\mathbf{X})$. Además debe calcularse la diferencia ΔE en la función de costo entre ambas soluciones:

$$\Delta E = f(\mathbf{X}) - f(\mathbf{X}_{act}) \quad [2.2]$$

Si $\Delta E < 0$ el costo de la nueva solución $\mathbf{X}$ es menor al costo de la solución actual $\mathbf{X}_{act}$) la nueva solución es aceptada, es decir, que una solución de menor costo siempre se acepta. En caso contrario, la nueva solución es aceptada con una probabilidad $P(\Delta E)$:

$$P(\Delta E) = e^{-(\Delta E / t)} \quad [2.3]$$

Obsérvese que la ecuación [2.3] es la misma ecuación [2.1] pero sin la constante de Boltzman (k), ya que ésta no se considera debido a que no tiene significado en los problemas de optimización (Dowsland y Díaz, 2001).

En la práctica se selecciona un número aleatorio entre 0 y 1, y si este número es menor que $P(\Delta E)$ la nueva solución se acepta.

Aceptar una solución de menor calidad permite salir de un posible mínimo local y explorar otras áreas del espacio de soluciones. Como la simulación comienza con una temperatura alta, $P(\Delta E)$ es cercana a 1 y por lo tanto una nueva solución con un costo mayor tiene una alta probabilidad de ser aceptada. La probabilidad de aceptar una solución peor va disminuyendo a medida que la temperatura decrece.

Para cada nivel de temperatura, el sistema debe alcanzar un equilibrio, es decir, un número de nuevas soluciones debe ser ensayado antes de que la temperatura sea reducida. Se puede mostrar que el algoritmo encontrará, bajo ciertas condiciones el mínimo global y no se estancará en un mínimo local (Moins, 2002).

El diagrama de flujo correspondiente al algoritmo básico de recocido simulado para minimización se presenta en la figura 2.1. El algoritmo para el caso de maximización puede ser fácilmente deducido.

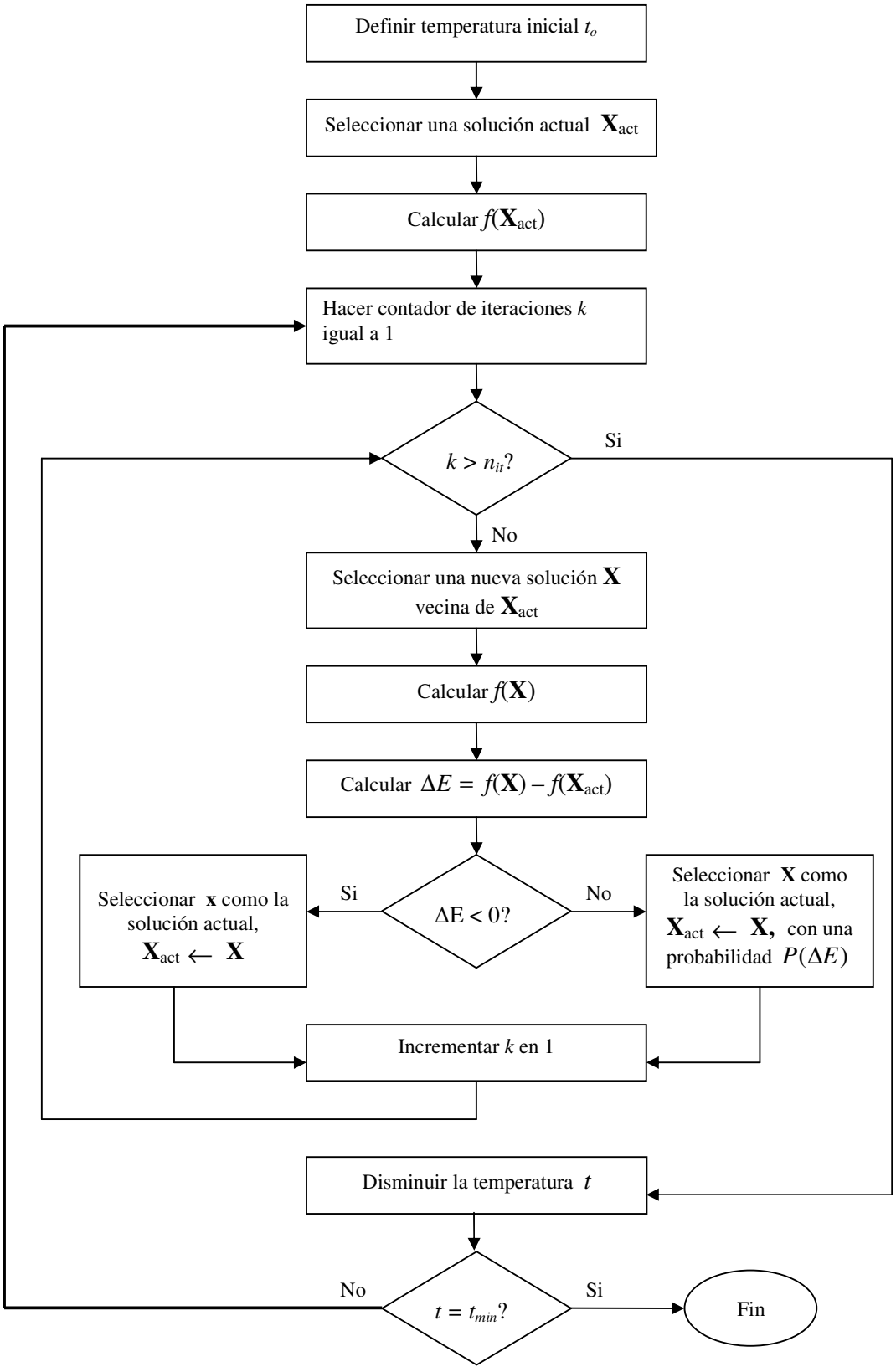


Figura 2.1 Pasos básicos del recocido simulado

El bucle interno genera una nueva solución en el entorno de la solución actual y la acepta si es mejor, en el caso de que esta nueva solución sea de menor calidad se acepta con una probabilidad $P(\Delta E)$. Este proceso se repite n_{it} veces. Cuando este ciclo iterativo se completa, la temperatura se disminuye y comienza nuevamente el bucle interno, repitiendo el proceso de creación, evaluación y posible aceptación de soluciones vecinas. Cuando la temperatura es lo suficientemente baja ($t = t_{min}$) el algoritmo finaliza y debe indicar cuál fue la mejor solución obtenida.

Para poder implementar el algoritmo de recocido simulado para resolver un problema concreto es necesario tomar ciertas decisiones referentes a la definición de parámetros (temperatura inicial t_o, número de iteraciones en cada nivel de temperatura n_{it}, temperatura mínima t_{min}), la tasa de disminución de la temperatura, el mecanismo para generar soluciones vecinas, entre otras. Estas decisiones se pueden clasificar en decisiones genéricas y específicas.

2.4 Decisiones genéricas

Las decisiones genéricas están relacionadas con los parámetros que rigen el programa de enfriamiento, incluyendo la temperatura inicial, la forma en que se disminuye la temperatura, número de iteraciones en cada nivel de temperatura y el criterio de parada del algoritmo (temperatura mínima). En ausencia de resultados teóricos generales, es necesario hacer ajustes empíricos de estos parámetros y para ciertos problemas, la tarea es complicada debido a la gran sensibilidad de los resultados al valor de dichos parámetros. En las siguientes secciones se presentan algunas consideraciones generales que pueden orientar las decisiones genéricas.

Temperatura inicial. El algoritmo debe comenzar con una temperatura alta que permita que muchos movimientos o soluciones vecinas sean aceptados. En la práctica, se requiere conocer el valor de la función de costo para las soluciones de vecinas, por ejemplo, si el mayor incremento en la función objetivo (ΔE) entre soluciones vecinas es conocido, sería posible calcular un valor t_o que aceptase un movimiento con cierta probabilidad usando la ecuación 2.3. En la ausencia de tal conocimiento, se puede seleccionar una temperatura que parezca ser un valor alto, ejecutar el algoritmo para un tiempo corto y observar la tasa de aceptación. Si esta tasa es "convenientemente alta", la temperatura con la cual se

experimentó puede ser usada como el valor inicial t_o. El significado de "convenientemente alta" varía de una situación a otra, pero en muchos casos una tasa de aceptación ente 40% y 60% parece dar buenos resultados (Reeves, 1996; Dowsland y Díaz, 2001). .

Disminución de temperatura. Se refiere a la forma de la curva de enfriamiento que se utiliza en el recocido simulado, la cual determina la velocidad de disminución de la temperatura a medida que avanzan las iteraciones del algoritmo.

Existen diferentes maneras de abordar el decrecimiento de la temperatura, una de las más utilizadas debido a su simplicidad y a los buenos resultados que ha dado en numerosas aplicaciones es la forma exponencial o geométrica:

$$t_{k+1} = \alpha\, t_k \qquad [2.4]$$

Donde t_{k+1} es la temperatura en la iteración $k+1$, t_k es la temperatura en la iteración k y α es una constante cercana a 1, escogida por lo general en el rango de 0.9 a 0.99.

Otro método bastante utilizado es el propuesto por Lundy y Mess (1986):

$$t_{k+1} = \frac{t_k}{1 + \beta \cdot t_k} \qquad [2.5]$$

Siendo β una constante cuyo valor está cerca de 0.

Se pueden utilizar otras formas funcionales en el programa de enfriamiento, sin embargo, no hay en la literatura recomendaciones concisas acerca de cuál es la mejor, puede depender del problema y en ese caso debe decidirse por experimentación.

Número de iteraciones. El recocido simulado ejecuta cierto número de iteraciones en cada nivel de temperatura para alcanzar el equilibrio, una vez alcanzado este estado la temperatura se reduce y el proceso se repite. Un esquema obvio es mantener un número de iteraciones constante en cada temperatura o alternativamente, se puede variar según desciende la temperatura, dedicando suficiente tiempo de búsqueda a temperaturas bajas para garantizar que se visita el óptimo local, es decir que conforme disminuye la temperatura se aumenta el número de repeticiones (Dowsland y Díaz, 2001).

El número de iteraciones en cada temperatura está relacionado con el tamaño del problema. De acuerdo a Voß (2001) el número de iteraciones puede ser calculado de la siguiente manera:

$$n_{it} = Factor\ x\ |N| \qquad [2.6]$$

Donde $|N|$ es el tamaño del entorno actual y *Factor* es un parámetro adecuado.

Dréo et al. (2006) sugieren que se puede cambiar a otro nivel de temperatura inferior cuando ocurra alguna de las siguientes condiciones:

- $12\ x\ G$ movimientos son aceptados
- $100\ x\ G$ movimientos evaluados

Donde G indica los grados de libertad (o parámetros del problema).

Otro enfoque relacionado al número de iteraciones consiste en reducir la temperatura una pequeña cantidad después de cada movimiento, es decir que el número de iteraciones en cada temperatura es igual a 1. Este es menos complicado y es el más utilizado en la práctica (Reeves, 1996).

Temperatura final. Teóricamente la temperatura debería reducirse hasta 0, pero en la práctica no es necesario debido a que la búsqueda converge por lo general a su óptimo local final antes de llegar a ese valor nulo de la temperatura. Además, el algoritmo puede hacerse demasiado largo y los tiempos de computación pueden ser muy grandes, especialmente si se utiliza una tasa de decrecimiento geométrica. Hasta cierto grado la temperatura final puede depender del problema específico, y como en el caso de la temperatura inicial, se puede hacer una estimación a partir de establecer la probabilidad de aceptación que se desea en la etapa final del algoritmo.

Un criterio diferente es detener la búsqueda cuando se haya producido un número determinado de iteraciones sin alguna aceptación. Así por ejemplo, Dréo et al. (2006) sugieren que se debe terminar el algoritmo después de tres etapas sucesivas de temperatura sin que haya habido alguna aceptación.

2.5 Decisiones específicas

Este conjunto de decisiones están relacionadas al problema específico que se desea resolver. Se refieren principalmente a la función de costo, la definición de la estructura de entornos y el espacio de soluciones.

Función de costo. Esta función debe medir la calidad de una solución y se define de tal manera que represente el problema que se está tratando de resolver. En los casos de optimización restringida, además de la función objetivo se incluyen en la función de costo las restricciones con ciertos pesos. Las restricciones "fuertes" o "duras"[1] son consideradas en la función de costo con un peso alto de tal manera que, una solución que viole estas restricciones tendría un valor de la función de costo mayor que una solución factible. Las restricciones "débiles" o "suaves" [2] dependiendo de su importancia pueden incluirse en la función de costo con un peso asociado.

Un aspecto que es importante tomar en cuenta relacionado con la función de costo es el cálculo de su valor para una solución determinada. En cada iteración del algoritmo cuando se halla una nueva solución es necesario calcular el cambio de la función de costo con respecto a la solución anterior. Es posible disminuir el tiempo de computación si este cambio se calcula usando información relativa a qué parte de la solución ha cambiado, sin proceder a evaluarla totalmente sin tener en cuenta la información previamente disponible (Dowsland y Díaz, 2001).

Estructura de entornos. La decisión más importante en un problema de recocido simulado es la definición de la estructura de entornos, o lo que es igual, establecer cómo pasar de una solución a otra solución vecina. Algunos de los primeros desarrollos teóricos se basaban en entornos simétricos, es decir, si se pasa de una solución i a una solución j entonces debe ser posible moverse de la solución j a la solución i (Gökçe, 2007). Resultados teóricos demuestran que es suficiente con exigir que cualquier solución pueda alcanzarse desde cualquier otra a través de una serie de movimientos válidos (Reeves, 1996, Dowsland y Díaz, 2001). Este requerimiento se denomina condición de "alcanzabilidad".

De esta manera, al momento de trabajar con un problema determinado es necesario asegurarse de que esta condición se cumpla.

---

[1] Restricciones que no pueden ser violadas porque originarían una solución infactible.
[2] Restricciones que no causan infactibilidad pero que se desean cumplir en la medida de lo posible.

Espacio de soluciones. Posee una topología que se origina en el concepto de proximidad entre dos soluciones (estructura de entornos). Existe un costo asociado a cada solución, de tal forma que el espacio de soluciones está caracterizado por un "paisaje de costo". La dificultad de un problema de optimización radica en que este paisaje comprende un gran número de valles de profundidad variable y que corresponden a mínimos locales (ver figura 2.2). La forma de este paisaje depende bastante de la función de costo y de la estructura de entornos.

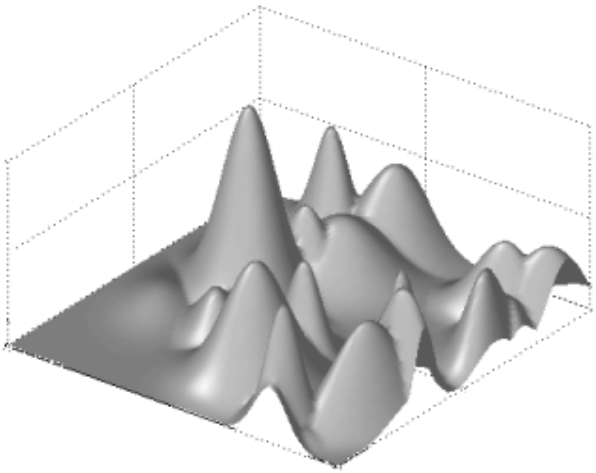


Figura 2.2 Paisaje de la función de costo

Problemas de optimización combinatoria resueltos con recocido simulado

La efectividad del recocido simulado fue probada inicialmente sobre algunos problemas clásicos de optimización combinatoria, tales como el problema del vendedor viajero, partición de grafos, mínimo apareamiento de puntos y asignación cuadrática. La comparación con otros algoritmos conducen a diferentes resultados, variando de acuerdo a los problemas y los autores. En algunos problemas de grandes dimensiones (cientos de variables) se observaron resultados prometedores con recocido simulado.

El recocido simulado se ha utilizado en diferentes áreas del conocimiento y aplicaciones, tales como el trazado de circuitos electrónicos, procesamiento de imágenes, robótica, planificación de horarios y programación de actividades, entre muchas otras.

2.7 Aplicaciones en planificación forestal

En el campo de la planificación forestal se ha utilizado para buscar soluciones a problemas complejos de optimización, tales como aquéllos en los que se incluyen restricciones de integridad, restricciones espaciales o de adyacencia, y problemas de transporte forestal. Puede mencionarse los trabajos de Lockwood y Moore (1992), Tarp y Helles (1997), Crowe y Nelson (2005), y Chen y Gadow (2002, 2008), quienes desarrollaron algoritmos de RS para solucionar problemas con restricciones espaciales. Dahlin y Sallnas(1993), Murray y Church(1995), Boston y Bettinger (1999), Heinonen y Pukkala (2004), Pukkala y Kurtila (2005), Liu et al (2006) compararon el RS con otras técnicas heurísticas en el ámbito forestal. Otros autores como Chung y Sessions (2003) y Rodrigues et al. (2004) resolvieron problemas de planificación forestal de diferentes dimensiones usando recocido simulado.

2.8 Ejemplo de aplicación

Se desea elaborar un plan de aprovechamiento para una unidad de producción forestal formada por seis rodales de una misma especie, para abastecer de madera en forma continua a una planta de pulpa para la producción de cartón. El objetivo es establecer una secuencia de corta que minimice los costos totales de aprovechamiento y satisfaga la demanda de la planta. El período de planificación es de cinco años y la cuota anual (demanda) a cumplir es de 20000 toneladas.

De cada rodal se conoce su edad, área e incremento medio anual (IMA), y a partir de estos datos se calcula el volumen disponible de madera en cada rodal para cada año del período de planificación. Se considera que el IMA disminuye en forma constante un 5% por año, después que la plantación alcanza y sobrepasa la edad del turno óptimo. El turno mínimo es a los 6 años y el turno óptimo a los 7 años para la especie considerada (eucalipto). En las tablas 2.2 y 2.3 se presentan los datos básicos de cada rodal y los datos de volumen.

Tabla 2.2 Datos de los rodales

Rodal	Edad (años)	Area (ha)	IMA (m3/ha/año)
1	7	245	29
2	4	78	26
3	6	305	18
4	8	154	27
5	5	208	27
6	2	310	26

Tabla 2.3 Volumen disponible en cada rodal

	Volumen (ton)				
Rodal	año 1	año 2	año 3	año 4	año 5
1	39788	43198,4	46168,29	48733,195	50926,1888
2	0	0	9734,4	11356,8	12330,24
3	26352	30744	33379,2	35674,02	37655,91
4	25280,64	27018,684	28519,722	29803,1095	30886,8589
5	0	26956,8	31449,6	34145,28	36492,768
6	0	0	0	0	12896

Se considera que el costo total de aprovechamiento tiene tres componentes: el costo de cosecha, el costo de transportar la madera a la planta y el costo de oportunidad (costo de aprovechar un rodal en una edad diferente a su turno óptimo). El costo de cosecha se define constante para todos los rodales e igual a 20 u.m./ton (unidades monetarias por tonelada). En las tablas 2.4, 2.5 y 2.6 se muestran los costos de transporte, los costos de oportunidad (calculados de acuerdo al procedimiento utilizado por Chiari et al., 2008) y los costos totales de aprovechamiento para cada rodal en cada año del período de planificación.

Tabla 2.4 Costos de transporte

Rodal	Costo de transporte (u.m/ton)
1	12
2	12
3	12
4	18,5
5	17
6	18,5

Tabla 2.5 Costos de oportunidad

	Costo de oportunidad (u.m./ha)				
Rodal	año 1	año 2	año 3	año 4	año 5
1	0	25,1	100,3	208,5	337,7
2	0	0	39,4	0	19,9
3	46	0	23,2	92,9	193,1
4	0	81,2	198,1	337,6	490,6
5	0	42,6	0	21,53	86
6	0	0	0	0	33,8

Tabla 2.6 Costo total de aprovechamiento

	Costo total (u.m.)				
Rodal	año 1	año 2	año 3	año 4	año 5
1	1273216	1388498,3	1501958,78	1610544,74	1712374,54
2	0	0	314574	363417,6	396119,88
3	857294	983808	1075210,4	1169903,14	1263884,62
4	973304,64	1052724,13	1128516,7	1199410,12	1264696,47
5	0	1172496	1263375,36	1354704,42	1767886,25
6	0	0	0	0	1048314,8

Las celdas con costo total igual a 0, se refieren a rodales que no pueden ser cortados en un año determinado debido a que no han alcanzado la edad mínima de corta.

Formulación matemática del modelo

Minimizar $Z = \sum_{i=1}^{6}\sum_{j=1}^{5} C_{ij} * X_{ij}$ [2.7]

Sujeta a

$$\sum_{i=1}^{6} V_{ij} * X_{ij} \geq 20000 \qquad j = 1, 2, \ldots,5 \qquad [2.8]$$

$$\sum_{j=1}^{5} X_{ij} = 1 \qquad i = 1, 2, \ldots,6 \qquad [2.9]$$

$$X_{ij} = (0, 1) \qquad i = 1, 2, \ldots, 6 \; ; \; j = 1, 2, \ldots,5 \qquad [2.10]$$

Donde:

C_{ij} = costo total (costo de cosecha + costo de transporte + costo de oportunidad) de cortar el rodal *i* en el año *j*;

V_{ij} = volumen total (toneladas) presente en el rodal *i* el año *j*;

X_{ij} = variable de decisión binaria, es igual a 1 si se corta el rodal *i* en el año *j*, y vale 0 en caso contrario.

El modelo tiene como función objetivo la minimización de los costos totales de aprovechamiento [2.7], sujeta a dos grupos de restricciones. El primer grupo [2.8] corresponde al cumplimiento de la cuota anual y el segundo grupo [2.9] son las restricciones de singularidad, que se refieren al hecho que un rodal sólo puede ser cortado en un año del período de planificación, la igualdad de estas restricciones indica que al final del período de planificación todos los rodales deben haber sido aprovechados. Si se desea plantear del modelo de tal manera que un rodal pueda ser o no aprovechado durante el período de planificación, estas restricciones deberían ser del tipo ≤.

Solución del modelo utilizando el algoritmo exacto de ramificación y acotamiento

A fin de realizar comparaciones posteriores, se resolvió el modelo haciendo uso el software de optimización SAS OR, el cual utiliza el algoritmo de ramificación y acotamiento para obtener la solución óptima. La secuencia óptima de cortas obtenida es la siguiente:

Tabla 2.7 Plan de cortas óptimo

Año	Rodal(es) a cortar	Volumen total aprovechado (ton)
1	**1**	**39788**
2	**5**	**26956,8**
3	**3**	**33379,2**
4	**4**	**29803,1**
5	**2 y 6**	**25226,2**

Costo mínimo: 5457192,8 u.m.

Solución del modelo utilizando recocido simulado

El problema se resolvió usando un algoritmo de recocido simulado, programado usando el lenguaje Visual Basic 6.0. En la implementación del algoritmo se aplicaron las siguientes estrategias:

*a. Representación de una solución***:** se hace mediante un vector de ceros y unos, cada elemento del vector almacena una variable binaria X_{ij} que indica si un determinado rodal *i* se cortará o no en el año *j*. El vector tiene tamaño n:

n = número de rodales x período de planificación

n= 6 x 5 = 30

En la figura 2.3 se muestra una representación de una posible solución.

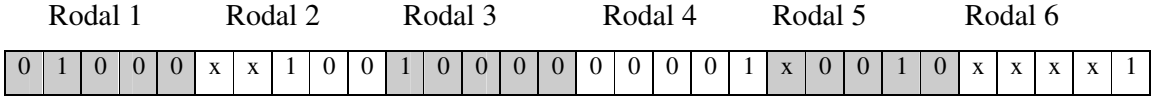


Figura 2.3 Representación de una solución

Se observa en la figura 2.3 que en los primeros cinco elementos del vector se guardan las variables relacionadas con la corta del rodal 1, los siguientes cinco elementos corresponden al rodal 2, y así sucesivamente. De esta manera, cada rodal tiene asociadas cinco posiciones del vector, correspondientes a los cinco años del período de planificación. La solución que

está representada en la figura 2.3 indica que el rodal 1 se corta en el año 2, el rodal 2 en el año 3, los rodales 4 y 6 en el año 5, y el rodal 5 en el año 4.

Las celdas marcadas con x se refieren a períodos de tiempo en los que no se puede aprovechar un rodal debido que no ha alcanzado la edad mínima de corta. Esas variables no se toman en cuenta, se incluyen en el vector para mayor facilidad en la programación y se les asigna un valor constante igual a cero que no cambia durante la ejecución del programa.

b. Espacio de soluciones: el total de variables para el problema es 22. Puede notarse en la figura 2.3 que el vector tiene tamaño 30, pero hay siete posiciones que no son variables (las marcadas con x). Además, el rodal 6 debe ser aprovechado el año 5 ya que antes no es posible debido a su edad y por los supuestos del problema debe ser aprovechado en el período de planificación, por lo tanto esa variable debe ser igual a 1 en cualquier solución, pudiéndose dejar de contar como variable ya que su valor va a ser fijo. De esta manera, el problema tiene 22 variables.

Si se considera que un rodal sólo puede aprovecharse una vez durante el período de planificación, se tiene que para los rodales 1, 3 y 5 hay 5 posibles años en los que se pudiera cortar, para el rodal 2 hay 3 posibilidades, el rodal 5 tiene 4, y el rodal 6 sólo tiene una (ser cortado en el año 5). Por lo tanto, existen 5 x 5 x 5 x 3 x 4 x 1 = 1.500 posibles soluciones.

c. Función de costo: se define como una función objetivo con penalizaciones, tal que incluye la función objetivo original más una penalización por la violación total (medida en toneladas) de las restricciones de demanda. El cumplimiento de las restricciones de singularidad se garantiza en los procedimientos utilizados para generar una solución inicial y las soluciones vecinas, por esta razón, no se incluyen esas restricciones en la función de costo.

$$f^{p}(\mathbf{X}) = f(\mathbf{X}) + P^{*} VT(\mathbf{X}) \qquad [2.11]$$

Donde:

$f^{p}(\mathbf{X})$: función de costo o función objetivo penalizada

$f(\mathbf{X})$: función objetivo del problema original

P : penalización que se impone a la violación de las restricciones de demanda (u.m./ton)

VT(**X**): violación total de las restricciones de demanda (ton) en la que incurre una solución **X**.

La violación total de las restricciones de demanda para una solución **X** está dada por:

$$VT(\mathbf{X}) = \sum_{k=1}^{5} (20000 - g_k) \cdot I_k \qquad [2.12]$$

Donde:

g_k : valor de la restricción de demanda para el año ***k*** evaluada en **X**, o lo que es igual, volumen aprovechado en el año *k* si se aplica el plan de cortas (solución) **X**.

I_k: variable indicadora que vale 1 si en el año *k* se viola la restricción de demanda, es decir, el volumen que se corta en ese año es menor a la cuota anual (g_{jk} ‹ 20000). En otro caso, vale 0.

d. Solución inicial: se genera aleatoriamente. Para cada rodal se elige al azar un año en el que será cortado (solamente se consideran los años en que puede ser cortado ese rodal) y se asigna 1 a las variables (posiciones del vector) correspondientes. A los demás variables representadas en el vector se les asigna 0. Con esta estrategia se asume que todos los rodales deben ser cortados durante el período de planificación.

e. Generación de soluciones vecinas: partiendo de una solución se genera una solución vecina de ésta, eligiendo aleatoriamente un rodal para modificar su año de corta. Luego, se elige un año para cortar dicho rodal, ese año debe ser diferente al año en que se corta ese rodal en la solución de partida. Por ejemplo: supóngase que se quiere obtener una solución vecina para la solución representada en la figura 2.3, siguiendo la estrategia antes descrita, se elige al azar el rodal 4 y el año 3, entonces la solución vecina sería la solución representada en la figura 2.4.

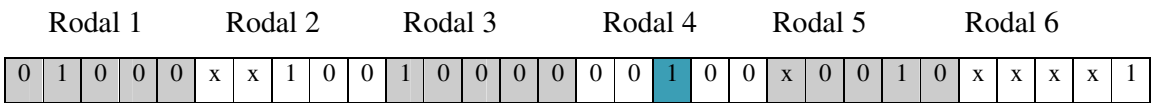


Figura 2.4. Solución vecina

f. Parámetros del algoritmo de recocido simulado: se definieron mediante experimentación.

- Temperatura inicial: 10 x 10^8
- Iteraciones del algoritmo: 100
- Iteraciones en cada nivel de temperatura: 50
- Tasa de disminución de la temperatura (α) : 0,9
- Valor de penalización (P): 1000

Resultados

Se hicieron 100 corridas del algoritmo de recocido simulado. En el 87% de las corridas alcanzó la solución óptima y el 13% fueron soluciones factibles cercanas al óptimo. El costo promedio obtenido fue 5459613,53 u.m..

Para tener una medida de la eficiencia del algoritmo se calculó en cada corrida el error relativo de acuerdo a la siguiente ecuación.

$$\text{Error relativo} = \frac{Vopt - Vrs}{Vrs} x\ 100\% \qquad [2.13]$$

Donde:

Vopt = valor objetivo óptimo

Vrs = valor objetivo obtenido en una corrida del algoritmo de recocido simulado.

El error relativo promedio obtenido para las 100 corridas fue de 0,04 %.

En cuanto al tiempo promedio de ejecución del algoritmo de recocido simulado, éste fue de 250,75 milisegundos, mientras que el tiempo de ejecución del algoritmo exacto de ramificación y acotamiento en SAS fue de 710 milisegundos.

CAPÍTULO III

Búsqueda Tabú (*Tabu Search*)

3.1 Origen de la Búsqueda Tabú

La búsqueda tabú es un método heurístico originalmente propuesto por Glover (1986) para resolver problemas de optimización combinatoria. Los principios fundamentales de la búsqueda tabú se fueron delineando en una serie de artículos publicados a finales de los años 80 y principios de los 90, y han sido unificados en el libro "*Tabu Search*" escrito por Glover y Laguna (1997).

Esta técnica al igual que el recocido simulado está basada en la búsqueda de soluciones vecinas evitando óptimos locales, pero lo hace en forma determinística emulando los procesos de memoria del ser humano. Incorpora inteligencia al utilizar el concepto de "memoria" para conducir la búsqueda a nuevas zonas del espacio de soluciones e impedir búsquedas repetidas en áreas ya exploradas.

3.2 Principios del método

El paradigma básico de la búsqueda tabú es utilizar información acerca de la historia reciente del proceso de búsqueda para superar el problema de la optimalidad local. El método memoriza las soluciones que han sido examinadas recientemente, denominándolas puntos tabú (prohibidos) y evita que estos puntos sean nuevamente considerados al seleccionar una próxima solución.

Tal como ocurre en la técnica de recocido simulado, la búsqueda puede conducir a aceptar movimientos que deterioran el valor objetivo, en este caso, se puede aceptar una solución peor cuando en un entorno no existen movimientos que mejoran la solución o cuando estos movimientos son considerados tabú. Esto ayuda a evitar ciclos y sirve además para diversificar la búsqueda de soluciones.

Según Glover (1990) la búsqueda tabú está basada en tres fundamentos:

1. El uso de estructuras flexibles de memoria diseñadas para almacenar la información histórica de la búsqueda.

2. Un mecanismo de control usado para emplear las estructuras de memoria, el cual incluye algunas condiciones que limitan la búsqueda y otras que la hacen más flexible. Este mecanismo se encuentra inmerso en la técnica en la forma de restricciones tabú y criterios de aspiración.

3. La incorporación de memorias de diferente duración (de corto y largo plazo), para implementar estrategias que intensifiquen y diversifiquen la búsqueda.

3.3 Algoritmo básico de búsqueda tabú

El algoritmo más sencillo de búsqueda tabú comienza con la selección de una solución de partida ($\boldsymbol{X}$) y la inicialización de una estructura de memoria (memoria de corto plazo), luego continúa un procedimiento iterativo de búsqueda de una mejor solución.

En cada iteración se define un conjunto de soluciones vecinas $N^*(\boldsymbol{X})$ que corresponde a un entorno reducido de la solución actual $\boldsymbol{X}$. No se evalúa el entorno completo de $\boldsymbol{X}$ sino un subconjunto de éste, $N^*(\boldsymbol{X}) \subset N(\mathbf{X})$. La razón por la que el entorno reducido es distinto al entorno estándar se debe a la memoria de corto plazo, también llamada lista tabú T (ver figura 3.1). Esta memoria restringe las soluciones que pueden alcanzarse desde una solución dada, ya que prohíbe ciertos movimientos. La memoria de corto plazo almacena las soluciones que se han visitado recientemente, de tal forma que quedan prohibidos los movimientos hacia esas soluciones. En estas condiciones, el entorno reducido se puede definir como $N^*(\boldsymbol{X}) = N(\mathbf{X}) \setminus T$ (Duarte et al., 2008).

La búsqueda tabú para una solución $\boldsymbol{X}$ evalúa todas las soluciones vecinas que pertenecen a $N^*(\boldsymbol{X})$ y selecciona la mejor de ellas, aún cuando esta solución desmejore el valor objetivo en relación a otras soluciones previamente encontradas. Si esta solución es mejor que la mejor solución encontrada hasta el momento, se establece como la nueva mejor solución encontrada. Al finalizar cada iteración la solución $\boldsymbol{X}$ se almacena en la memoria o lista tabú.

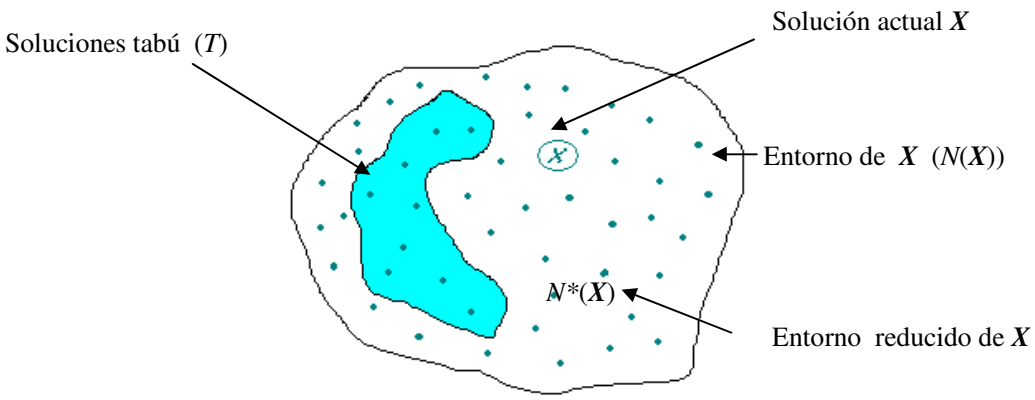


Figura 3.1 Representación de la estructura de entorno de una solución

De esta manera, la búsqueda tabú en forma iterativa va de una solución a otra hasta que se satisfaga un criterio de parada, el cual pudiera ser alguno de los siguientes (Hertz et al., 1995):

- El entorno reducido $N^*(\boldsymbol{X})$ es un conjunto vacío (puede ser que todas las soluciones vecinas sean tabú).
- Se ha superado un número máximo de iteraciones.
- Se considera que la solución obtenida es cercana a la solución óptima esperada.
- Ha pasado un cierto número de iteraciones sin que haya habido una mejora en la función objetivo.

En la figura 3.2 se muestra un diagrama de flujo de la búsqueda tabú. En el algoritmo básico se hace uso de la memoria de corto plazo, la cual conforma el corazón de la búsqueda tabú; los detalles para la implementación de este tipo de memoria se presentan en la siguiente sección. En versiones mejoradas del algoritmo se puede incluir una estructura de memoria de largo plazo, con estrategias de intensificación de la búsqueda en un área promisoria del espacio de soluciones y diversificación hacia otras zonas no exploradas.

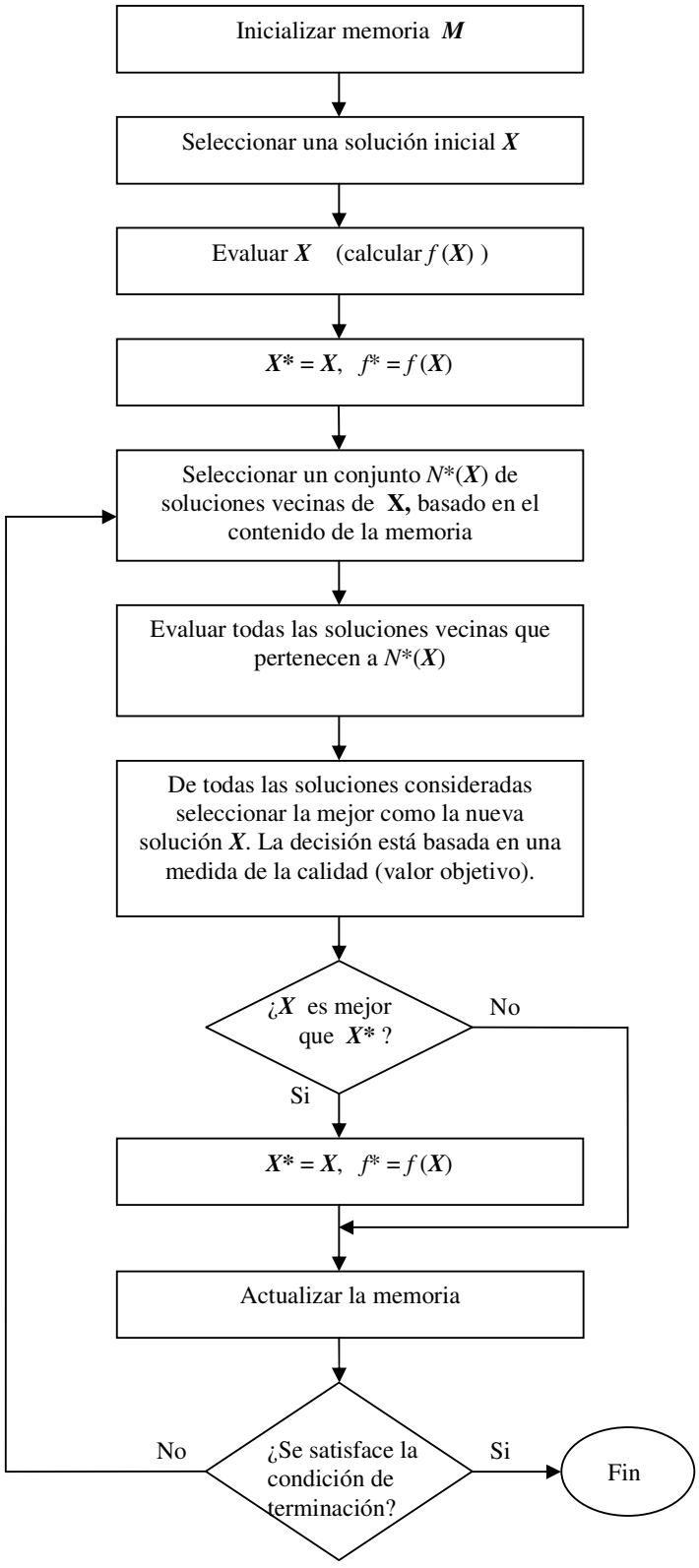


Figura 3. 2. Pasos básicos de la Búsqueda Tabú

3.4 Lista tabú (memoria de corto plazo)

En su forma más simple, la lista tabú es una memoria de corto plazo que contiene las soluciones que fueron visitadas en el pasado reciente. Tal como se señaló anteriormente, la búsqueda tabú etiqueta a las soluciones recientes como "tabú" para evitar visitarlas nuevamente y conducir la búsqueda hacia otras regiones del espacio de soluciones, previniendo efectos cíclicos en la búsqueda. De esta manera, la lista tabú es la estructura de memoria utilizada para almacenar las soluciones tabú (prohibidas) en una iteración del algoritmo.

Hay varias maneras de implementar una lista tabú. La forma más obvia es mantener una lista de todas las soluciones previamente visitadas y definir como tabú cualquier movimiento que pueda conducir a una solución de esta lista. Este enfoque requiere de capacidades de almacenamiento considerables y tiempos de computación altos para chequear si un posible movimiento es tabú, puede ser problemático aún cuando sólo se almacenen las soluciones más recientes (Reeves, 1996).

La manera más comúnmente utilizada para definir la lista tabú es almacenar los últimos movimientos realizados y prohibir los movimientos reversos. Otros enfoques para implementar la lista tabú están basados en características de las soluciones o de los movimientos, llamadas atributos. Se prohíbe que soluciones que tienen ciertos atributos sean consideradas y en vez de memorizar soluciones, se memorizan los atributos de las soluciones recientemente visitadas, por ejemplo, Atributos tabú = (0 * * * * * 1). Estos atributos se denominan *tabú - activos*. Las posibles soluciones que contengan elementos *tabú – activos* son "tabú".

De manera general, puede decirse que la memoria de corto plazo prohíbe realizar algunos movimientos, ya sea de manera directa almacenando movimientos o soluciones tabú, o indirectamente almacenando atributos de las soluciones que son prohibidas.

Después de cierto tiempo, las soluciones o movimientos almacenados en la lista pierden su estatus de tabú y son eliminados, ya que se supone que después de un determinado número de iteraciones la búsqueda estará suficientemente lejos y no será posible alcanzar nuevamente esa solución (Duarte et al., 2008). Debido a que la lista tabú sólo almacena los

movimientos más recientes, a la memoria de corto plazo se le denomina también "memoria basada en lo reciente".

Duración de las condiciones tabú. El número de de iteraciones que una solución, movimiento o atributo permanece en condición de tabú es un parámetro denominado "período tabú" (*t*: *tabu tenure*), el cual puede ser un número fijo o puede variar en el transcurso de la búsqueda.

En las aplicaciones estándar de búsqueda tabú se utiliza por lo general un valor fijo para el período tabú. En esos casos, las listas tabú con frecuencia son implementadas como una lista de longitud fija l, en la cual al añadir una nueva solución (movimiento) a la memoria se extrae la que se incluyó hace l iteraciones. De esta manera, la longitud de la lista corresponde al período tabú, es decir, $t = l$ (Duarte et al., 2008).

Supóngase que se tiene un problema de minimización, que puede ser representado como un paisaje limitado por un territorio que define las soluciones factibles y que está conformado por montañas y valles, donde su altitud corresponde al valor de la función objetivo. En esta situación, el efecto de la memoria de corto plazo es visitar un valle y algunas veces cruzar hacia otros valles distintos. Con un número alto de movimientos tabú (l grande) es más probable visitar otros valles sin que se explore suficientemente el valle donde se encuentra. Por el contrario, si los movimientos son prohibidos durante pocas iteraciones habrá menos oportunidades de pasar a otros valles de los alrededores, porque es casi seguro que habrá un movimiento permitido que conducirá a una solución cercana a lo más profundo del valle actual (Dréo et al., 2006).

Más formalmente, para longitudes pequeñas de la lista tabú (pocos movimientos tabú) la búsqueda se concentrará en áreas pequeñas del espacio de búsqueda y puede tender a visitar las mismas soluciones una y otra vez. En cambio, longitudes grandes obligan a que se visiten otras zonas del espacio de búsqueda y la oportunidad de visitar varias buenas soluciones es mayor. Sin embargo, el número de movimientos tabú no debería ser muy grande, porque sería menos probable encontrar un buen óptimo local por falta de movimientos disponibles, en ese caso la búsqueda estaría dirigida por los pocos movimientos permitidos y no por el valor de la función objetivo.

Para aprovechar las ventajas de longitudes de la lista tabú grandes y pequeñas, en implementaciones más avanzadas de búsqueda tabú se permite que la longitud de la lista o período tabú varíe en forma dinámica a lo largo de la ejecución del algoritmo. En tal sentido, varias metodologías pueden utilizarse, por ejemplo, la longitud de la lista se puede seleccionar en forma aleatoria dentro de un intervalo preestablecido. En algunas aplicaciones estas estrategias demostraron ser más eficientes que las listas tabú de tamaño fijo.

Glover y Melián (2003) proponen algunas reglas ilustrativas para determinar el período tabú, clasificándolas en estáticas y dinámicas.

Reglas estáticas: elegir el período tabú t como una constante tal que $t = 7$ o $t = \sqrt{n}$, donde n es una medida de la dimensión del problema. Estos valores se han utilizado en diferentes aplicaciones y han dado buenos resultados, sin embargo, el valor de t debería ser establecido por experimentación para un problema particular.

Reglas dinámicas: pueden ser un valor simple o atributo dependiente.

- Dinámico simple: elegir t para variar (aleatoriamente o mediante un patrón sistemático) entre cotas t_{min} y t_{max}, tal que $t_{min} = 5$ y $t_{max} = 7$, o $t_{min} = 0.9\sqrt{n}$ y $t_{max} = 1.1\sqrt{n}$.
- Dinámico atributo dependiente: elegir t como en la regla dinámica simple pero determinar t_{min} y t_{max} de tal forma que sean mayores para aquellos atributos que son más atractivos (pueden estar basados en estrategias de intensificación, las cuales se detallan más adelante).

Ejemplo sencillo. Para clarificar el funcionamiento de la memoria de corto plazo se presenta un ejemplo corto adaptado de Reeves (1996), en éste se pueden apreciar los fundamentos básicos de este tipo de memoria.

Se considera el siguiente problema:

Maximizar $f(x) = x^3 - 60\,x^2 + 900\,x + 100$

donde x es codificada como un entero binario de 5 bits en el rango [0, 31]. Esta función tiene un máximo en (0 1 0 1 0), que corresponde a $x = 10$ ($f(x) = 4100$).

Para comenzar con la búsqueda tabú se selecciona como solución inicial (1 1 0 0 1), es decir x = 25 ($f(x)$ = 725).

Una solución vecina se obtiene mediante el cambio de un bit de la solución actual. Así, cada solución tiene un entorno de tamaño 5. Se utiliza un período tabú de 3 iteraciones.

Iteración 1

Solución actual: (1 1 0 0 1) $f(x)$ = 725

Vecinos:

Bit modificado	Solución	$f(x)$	
1	0 1 0 0 1	4069	*
2	1 0 0 0 1	2973	
3	1 1 1 0 1	129	
4	1 1 0 1 1	343	
5	1 1 0 0 0	964	

Se observa en esta iteración que de todos los vecinos el que tiene un mejor valor objetivo es la solución obtenida al cambiar el bit 1 (0 1 0 0 1), en consecuencia se selecciona como solución actual. Modificar el bit 1 será un movimiento tabú en las siguientes tres iteraciones.

La memoria a corto plazo o lista tabú se puede representar en este caso como un vector de cinco elementos (de igual tamaño que la solución) y se utiliza para guardar el número de iteraciones que un determinado bit permanecerá como tabú. Antes de comenzar la búsqueda esta memoria es un vector de 0´s, esto es, $\boldsymbol{M}$ = (0 0 0 0 0).

Al finalizar la primera iteración se actualiza el contenido del vector $\boldsymbol{M}$, quedando de la siguiente manera:

$\boldsymbol{M}$ = (3 0 0 0 0)

↓

Este valor significa que cambiar el bit 1 es un movimiento tabú por tres iteraciones

Iteración 2

Solución actual: (0 1 0 0 1) $f(x)$ = 4069

Vecinos:

Bit modificado	Solución	$f(x)$	
1	TABU		
2	0 0 0 0 1	941	
3	0 1 1 0 1	3857	
4	0 1 0 1 1	4071	*
5	0 1 0 0 0	3972	

La solución obtenida al modificar el bit 4 corresponde al mejor vecino, cambiar este bit será un movimiento tabú hasta la iteración 5.

El vector de memoria toma los siguientes valores:

$\boldsymbol{M}$ = (2 0 0 3 0)

El bit 4 será tabú en las 3 siguientes iteraciones

El bit 1 sigue siendo tabú por 2 iteraciones más.

Iteración 3

Solución actual: (0 1 0 1 1) $f(x)$ = 4071

Vecinos:

Bit modificado	Solución	$f(x)$	
1	TABU		
2	0 0 0 1 1	2287	
3	0 1 1 1 1	3475	
4	TABU		
5	0 1 0 1 0	4100	*

Se observa que en esta iteración el procedimiento de búsqueda tabú encuentra el valor óptimo del problema, sin embargo, el algoritmo no sabe que esta es la solución óptima y continúa explorando.

Se actualiza la memoria $\boldsymbol{M}$ = (1 0 0 2 3).

Iteración 4

Solución actual: (0 1 0 1 0) $f(x)$ = 4100

Vecinos:

Bit modificado	Solución	$f(x)$	
1	TABU		
2	0 0 0 1 0	1668	
3	0 1 1 1 0	3684	*
4	TABU		
5	TABU		

El mejor vecino en la iteración 4 desmejora el valor objetivo obtenido en la iteración anterior, no obstante, se acepta como nueva solución actual.

El vector de memoria después de esta iteración es $\boldsymbol{M}$ = (0 0 3 1 2).

El bit 1 ya no será tabú
para la próxima iteración

Iteración 5

Solución actual: (0 1 1 1 0) $f(x)$ = 3684

Vecinos:

Bit modificado	Solución	$f(x)$	
1	1 1 1 1 0	100	
2	0 0 1 1 0	3556	*
3	TABU		
4	TABU		
5	TABU		

La memoria actualizada es $\boldsymbol{M}$ = (0 3 2 0 1).

El proceso puede continuar dependiendo del criterio de parada del algoritmo. Se observa que la búsqueda puede no permanecer cerca del óptimo.

Criterios de aspiración. En la búsqueda tabú algunas de las soluciones prohibidas en una iteración determinada podrían ser de excelente calidad y no ser visitadas. Para mitigar este problema, se utilizan los llamados criterios de aspiración, éstos permiten ignorar el estado de tabú de una solución, si se cumplen algunas condiciones. Los criterios de aspiración añaden flexibilidad a la búsqueda al permitir la introducción de soluciones tabú al entorno reducido $N^*(\boldsymbol{X})$ de una solución.

El criterio de aspiración más simple y mayormente utilizado consiste en permitir un movimiento tabú si el valor objetivo de la solución resultante es mucho mejor que las soluciones hasta ese momento encontradas. Otros criterios de aspiración más complicados han sido propuestos en la literatura pero son raras veces utilizados (Gendreau, 2003).

3.5 Memoria de largo plazo (memoria basada en frecuencias)

En muchas aplicaciones los componentes de la memoria de corto plazo son suficientes para producir soluciones de alta calidad. Sin embargo, la búsqueda tabú llega a ser mucho más efectiva cuando se incluye memoria de largo plazo y sus estrategias asociadas, principalmente en la resolución de problemas "duros". Por otra parte, la posibilidad de encontrar mejores soluciones a medida que las iteraciones del algoritmo van avanzando (cuando la solución óptima no se ha alcanzado) es mayor cuando se utiliza memoria de largo plazo además de la memoria de corto plazo (Glover y Laguna, 1997).

Una forma fácil de implementar un procedimiento de búsqueda tabú es diseñar y probar primero el componente de memoria de corto plazo y después incorporar la memoria de largo plazo y sus estrategias si se necesitan refinamientos adicionales (Glover, 1990).

Si algunos movimientos son elegidos más frecuentemente que otros, se puede asumir que la búsqueda tiene problemas en explorar soluciones de composición variada y esto puede hacer que permanezca confinada a un área del espacio de soluciones. En estas situaciones, es posible que utilizar la memoria de corto plazo para prohibir movimientos cuyo reverso fue aplicado recientemente no sea suficiente para escapar de un determinado valle del paisaje de soluciones, por lo que se hace necesario introducir otros mecanismos de memoria para dirigir efectivamente la búsqueda a largo plazo.

La memoria de largo plazo proporciona un tipo de información que complementa a la proporcionada por la memoria de corto plazo, ampliando la base para seleccionar movimientos. Este tipo de memoria se basa en medidas de frecuencia definidas como proporciones, cuyos numeradores representan cuentas del número de ocurrencias de un evento particular (por ejemplo, el número de veces que un atributo particular pertenece a una solución o movimiento) y cuyos denominadores generalmente representan uno de cuatro tipos de cantidades:

1) El número total de ocurrencias de todos los eventos representados por los numeradores, tal como el número de iteraciones asociadas.

2) La suma de los numeradores

3) El máximo valor del numerador

4) La media del valor del numerador.

Los denominadores 3) y 4) dan lugar a lo que se puede llamar frecuencias relativas. En los casos en los que los numeradores representan cuentas ponderadas, algunas de las cuales pueden ser negativas, los denominadores 3) y 4) se expresan como valores absolutos y el denominador 2) se expresa como una suma de valores absolutos.

Su pueden identificar dos tipos de medidas de frecuencia: medidas de residencia y medidas de transición. La primera de ellas está relacionada al número de veces que un atributo es observado, es decir, identifica el número de veces que el atributo $X_j = p$ reside en un conjunto de soluciones obtenidas. Por otra parte, las medidas de frecuencia de transición se refieren al número de veces que un atributo X_j cambia de un valor p a otro valor q (Reeves, 1996; Glover y Melián, 2003).

Los atributos que tienen mayores medidas de frecuencia, al igual que los movimientos que fueron ejecutados recientemente, pueden iniciar un estado tabú activo. Aunque esta puede ser una manera de incluir la información que genera la memoria a largo plazo, el enfoque más empleado es penalizar los movimientos más frecuentemente aplicados, así, se modifica la función objetivo y algunos movimientos se hacen más atractivos que otros.

Varios métodos de penalización pueden ser utilizados, por ejemplo, la prohibición de realizar movimientos cuya frecuencia de ocurrencia durante la búsqueda exceda un límite determinado, o la adición de una penalidad proporcional a la frecuencia de un movimiento en la función objetivo. Este último método es beneficioso para problemas en los que la función objetivo toma sólo un pequeño número de valores, lo cual puede generar dificultad para dirigir la búsqueda ya que algunos vecinos pueden tener el mismo valor objetivo. Al usar penalidades, la búsqueda tenderá a elegir aquellos movimientos que se han empleado menos (Dréo et al., 2006).

En la actualidad, la mayoría de las aplicaciones utilizan un múltiplo lineal simple de una medida de frecuencia para crear un término de penalización, aunque también pudieran usarse otras funciones como el cuadrado de la frecuencia.

En cuanto a la implementación de los dos tipos de memoria existe cierta similitud entre ambas. En la memoria de corto plazo se puede almacenar el número de iteraciones que un movimiento permanecerá como tabú, así como en el ejemplo que fue explicado en la sección *lista tabú*, o también es posible almacenar la iteración en la cual se podrá utilizar nuevamente un movimiento; mientras que en la memoria de largo plazo se registra el número de veces que un movimiento fue seleccionado.

Para el caso del ejemplo sencillo de la sección *lista tabú*, en el que se maximiza la función $f(x) = x^3 - 60\,x^2 + 900\,x + 100$, se puede utilizar un vector que almacene en la posición j la cantidad de iteraciones en las que el bit j fue modificado. De esta manera, se puede definir un vector $\boldsymbol{H}$ que corresponde a la memoria de largo plazo, se inicializa en $\boldsymbol{H} = (0\ 0\ 0\ 0\ 0)$. El vector $\boldsymbol{H}$ cambia de la siguiente manera en el transcurso de las iteraciones del algoritmo:

Iteración	Memoria $\boldsymbol{H}$	
1	(1 0 0 0 0)	Se modifica el bit 1
2	(1 0 0 1 0)	Se modifica el bit 4
3	(1 0 0 1 1)	Se modifica el bit 5
4	(1 0 1 1 1)	Se modifica el bit 3
5	(1 1 1 1 1)	Se modifica el bit 2

Al final de la iteración 5, todos los bits se han modificado una vez. Si continúan haciendo más iteraciones, los elementos del vector $\boldsymbol{H}$ irán aumentando su valor de acuerdo a la frecuencia con que se van cambiando los diferentes bits. Así por ejemplo, después de 20 iteraciones se podría tener un vector de memoria a largo plazo como el siguiente $\boldsymbol{H} = (2\ 5\ 6\ 5\ 2)$.

3.6 Estrategias avanzadas

Hay dos estrategias que pueden utilizarse en los algoritmos de búsqueda tabú para obtener implementaciones más efectivas, sobre todo cuando se trata de problemas de difícil solución. Estas estrategias son intensificación y diversificación, de alguna manera ya se encuentran implícitas en la memoria de corto y de largo plazo, sin embargo, se pueden implementar con mayores detalles y otros mecanismos más agresivos en diseños avanzados.

La intensificación tiene como finalidad reforzar la búsqueda en áreas promisorias del espacio de soluciones, y la diversificación consiste en dirigir la búsqueda hacia zonas del espacio de soluciones que no han sido visitadas. Algunos detalles sobre las estrategias de intensificación y diversificación son analizadas en las siguientes secciones.

1) ***Intensificación.*** La idea fundamental detrás de esta estrategia es realizar una exploración más completa de las porciones del espacio de búsqueda que parecen más "prometedoras" para estar seguro que las mejores soluciones de estas áreas fueron visitadas. La intensificación se basa en la modificación de las reglas de escogencia para garantizar combinaciones de movimientos que históricamente han dado buenos resultados.

Cada cierto tiempo, el procedimiento normal de búsqueda se puede interrumpir para ejecutar una fase de intensificación. Una forma simple de llevar a cabo la intensificación es regresar a la mejor solución encontrada y reiniciar la búsqueda. También se puede implementar una manera de identificar las mejores soluciones (conjunto de soluciones élite) y regresar la búsqueda hacia estas soluciones, es posible establecer un valor de referencia relacionado al mejor valor objetivo obtenido que permitirá determinar si una determinada solución puede formar parte del conjunto élite (Glover y Laguna, 1997).

Otra manera de realizar la intensificación está basada en alguna estructura de memoria intermedia, en la que se registra el número de iteraciones consecutivas que algunos atributos han estado presente en la solución actual. Estos atributos comunes pueden servir para guiar la búsqueda de nuevas soluciones hacia espacios en donde existan tales atributos. (Gendreau, 2003).

La intensificación es usada en diversas implementaciones de búsqueda tabú, pero no siempre es necesaria, ya que en muchas situaciones el proceso de búsqueda normal es suficiente. En esos casos no es necesario gastar más tiempo explorando cuidadosamente regiones del espacio de búsqueda que ya han sido visitadas.

2) ***Diversificación.*** Es una estrategia que trata de solventar el problema de quedarse atrapado en una zona del espacio de soluciones, forzando la búsqueda en áreas inexploradas. En búsqueda tabú, la diversificación se logra hasta cierto punto mediante ciertas funciones de la memoria de corto plazo, pero es principalmente reforzada por la memoria de largo

plazo. La diversificación se basa en la modificación de las reglas de escogencia para incorporar a las soluciones atributos que no han sido frecuentemente usados en el proceso de búsqueda.

La diversificación se puede implementar mediante diferentes técnicas como penalizar movimientos muy utilizados en el pasado, construir soluciones candidatas compuestas por atributos poco utilizados para continuar la búsqueda, o forzar aquellos atributos de la solución actual (o de la mejor solución encontrada) que se han cambiado muy poco y comenzar de nuevo la búsqueda a partir de ese punto (diversificación por reinicio), entre otras. La estrategia de diversificación es particularmente útil cuando se pueden alcanzar mejores soluciones solamente por cruzar valles o montañas en la topología del espacio de soluciones (Glover y Laguna, 1997).

Es importante señalar que la selección de estrategias de diversificación es probablemente la decisión más crítica en el diseño de algoritmos heurísticos de búsqueda tabú, por lo que debe hacerse con mucho cuidado.

3.7 Elementos a considerar al implementar una búsqueda tabú

Al igual que en otros procedimientos de búsqueda como el recocido simulado, antes de implementar una búsqueda tabú se deben tomar decisiones relacionadas a la representación de la solución, la cual permite generar un espacio de soluciones y estructuras de entorno para un problema particular; una función de evaluación o función objetivo que mide la calidad de las soluciones; y un mecanismo de movimientos que permite definir la estructura de entornos.

Además, hay otros factores que deben ser considerados al diseñar un algoritmo de búsqueda tabú tales como:

- Naturaleza del período tabú (fija o dinámica) y determinación de sus valores asociados.
- Criterios de aspiración que se usarán para eliminar estados tabú.
- Estructura de la memoria de corto plazo.
- Estructura de la memoria de largo plazo y selección de un método de penalización.
- Estrategias que se utilizarán para intensificación, en caso de ser necesarias.
- Estrategias de diversificación, si se decide utilizarlas.

3.8 Problemas de optimización combinatoria resueltos con búsqueda tabú

La búsqueda tabú ha tenido innumerables aplicaciones en muy variadas áreas dentro de la investigación de operaciones durante los últimos años. En general, las aplicaciones más frecuentes encontradas en la literatura caen en alguna de las siguientes categorías:

- Planificación (por ejemplo: planificación de horarios, de máquinas, de fuerza de trabajo).
- Diseño (por ejemplo: diseño de redes de transporte, planeación de espacios arquitectónicos, diseños de redes tolerantes a fallas).
- Localización y asignación (por ejemplo: asignación cuadrática, asignación generalizada multinivel, planificación de distribución en planta).
- Lógica e Inteligencia Artificial (por ejemplo: reconocimiento y clasificación de patrones, integridad de datos, diseño y entrenamiento de redes neuronales).
- Tecnología (por ejemplo: inversión sísmica, construcción de estaciones espaciales, diseño estructural)
- Telecomunicaciones (por ejemplo: asignación de rutas, diseño de redes de servicio, arquitecturas inmune a fallas).
- Producción, inventario e inversión (por ejemplo: manufactura flexible, sistemas de producción "justo a tiempo", planeación de inventarios).
- Diseño de rutas (por ejemplo: rutas de vehículos, rutas de flotas mixtas, el problema del viajero).
- Optimización de gráficas (por ejemplo: partición de gráficas, coloreado de gráficas)
- Problemas de optimización combinatoria en general (por ejemplo, programación binaria, programación no lineal no convexa, optimización entera mixta).

En general, la búsqueda tabú ha tenido mucho éxito en problemas de optimización combinatoria. Glover (1990) afirma haber resuelto problemas de planificación cuya formulación mediante programación entera involucra entre uno y cuatro millones de variables en 22-24 minutos, usando sólo una computadora personal, y obteniendo resultados que se encuentran dentro del 98% de límite superior de optimalidad. Tan sólidos resultados

la hacen ver como una técnica de búsqueda muy prometedora dentro de la investigación de operaciones.

3.9 Aplicaciones en planificación forestal

En el campo de la planificación forestal también se ha utilizado con éxito esta técnica, en la literatura existen algunos trabajos en los que se usa búsqueda tabú para resolver problemas de optimización combinatoria dentro del manejo forestal. Entre los primeros modelos de planificación que utilizaron esta metaheurística se encuentra el desarrollado por Bettinger et al. (1997), quienes implementaron un algoritmo de búsqueda tabú para planificar cortas sujeto a restricciones de producción fija, restricciones de adyacencia y objetivos ecológicos de mantenimiento de la calidad del habitat para la fauna. Boston y Bettinger (1999) compararon la eficiencia de la búsqueda tabú con la programación entera Monte Carlo y recocido simulado, los resultados mostraron que la búsqueda tabú proporcionó mejores soluciones para cierto tipo de problemas; Brumelle et al. (1998) aplicaron búsqueda tabú para resolver problemas con restricciones de adyacencia y un enfoque multicriterio; Laroze y Gleber (1997) y Laroze (1999) aplicaron búsqueda tabú para optimizar patrones de corte de trozas a nivel de bosque.

Más recientemente, Richards y Gunn (2000, 2003) utilizaron búsqueda tabú para resolver problemas de optimización de cosechas y diseño de caminos forestales; Rodrigues et al. (2003) resolvieron mediante un algoritmo de búsqueda tabú cinco problemas de planificación forestal que involucraron entre 93 y 423 variables con restricciones enteras; Aruga et al. (2005) también emplearon búsqueda tabú y algoritmos genéticos para diseñar caminos forestales y obtuvieron buenas soluciones en un tiempo de computación razonable; Díaz et al. (2007) proponen una solución basada en búsqueda tabú para resolver un problema combinado de localización de maquinaria forestal para transportar trozas y el diseño de una red de rutas de acceso entre los puntos de localización de la maquinaria y los caminos existentes; Bettinger et al. (2007) usaron tres variantes de búsqueda tabú para resolver problemas de planificación forestal a nivel de paisaje con selección de actividades para un conjunto de rodales.

3.10 Ejemplo de aplicación

Considérese el mismo ejemplo de aplicación del capítulo anterior, en el que se quiere obtener un plan de aprovechamiento de costo mínimo para una unidad de producción forestal formada por seis rodales de una misma especie, que cumpla con una cuota anual de producción y satisfaga restricciones de singularidad.

Este problema se resolvió usando un algoritmo de búsqueda tabú implementado en el lenguaje de programación Visual Basic 6.0. El algoritmo se desarrolló bajo las siguientes estrategias:

a. Forma de representación de una solución, generación de la solución inicial y de las soluciones vecinas: estos aspectos fueron manejados de igual manera que en el algoritmo de recocido simulado (ver capítulo II).

b. Función objetivo: se utilizó la misma función de costo definida en el algoritmo de recocido simulado, esto es, una función objetivo con penalizaciones que incluye la función de costo total de aprovechamiento original más una penalización por la violación total (medida en toneladas) de las restricciones de demanda (ver ecuaciones 2.11 y 2.12).

c. Tamaño del conjunto de soluciones vecinas evaluadas en una iteración: es necesario definir si en una iteración se explota la vecindad total de la solución actual o una vecindad parcial. En este problema, es fácil verificar que una solución particular tiene 17 soluciones vecinas de acuerdo al esquema de vecindad definido, como es una cantidad pequeña se decide explotar la vecindad completa.

d. Memoria de corto plazo: se prohíben los movimientos recientemente empleados. Por ejemplo, supóngase que en la iteración k se seleccionó como mejor solución vecina del entorno la solución en la que se modifica el año de cosecha del rodal i, entonces modificar ese rodal será un movimiento tabú por t iteraciones.

La memoria de corto plazo se implementó como un vector de 5 elementos, cada uno de los cuales corresponde a un rodal [3]. El valor guardado en la posición i del vector es el número

[3] Obsérvese que no se incluye una casilla para el rodal 6, esto se debe a que este rodal debe cortarse necesariamente en el año 5 (único año en el que se puede cortar debido a su edad). Así, no es factible cambiar su año de corta en una solución vecina y por lo tanto no hay movimientos tabú asociados.

de iteraciones durante las cuales no puede modificarse el año de cosecha del rodal i, por ser un movimiento tabú. Estos valores se actualizan en cada iteración.

Por ejemplo, en una iteración determinada del algoritmo la memoria pudiera tener la siguiente forma:

0	0	1	0	2

En este caso particular, cualquier solución en la que se modifique el año de corta del rodal 3 es tabú por una iteración, y cualquier solución en la que se modifique el año de corta del rodal 5 es tabú por dos iteraciones.

e. Permanencia tabú (t): se utiliza un valor fijo de la permanencia tabú, siguiendo el criterio $t = \sqrt{n}$, donde n es una medida de la dimensión del problema. Si se considera que hay 5 rodales (el rodal 6 no se toma en cuenta ya que su año de corta no puede cambiar), t se puede calcular como $\sqrt{5} = 2.2$. Así, para este problema se tomó $t = 2$, un movimiento tabú permanece en esta condición por 2 iteraciones.

f. Criterios de aspiración: se aplica el criterio de aspiración más sencillo, sólo se acepta una solución tabú como solución actual en una iteración , si su valor objetivo es mejor que el valor objetivo de la mejor solución vecina (no tabú) del entorno y mejor que el valor objetivo de la mejor solución encontrada hasta el momento. Para utilizar este criterio de aspiración es necesario evaluar en cada iteración todas las soluciones vecinas tabú.

g. Memoria de largo plazo y estrategias avanzadas de intensificación y diversificación: no fue necesario incorporar estos elementos en el algoritmo, ya que el procedimiento básico de búsqueda tabú con memoria de corto plazo dio resultados bastante satisfactorios.

h. Parámetros del algoritmo de búsqueda tabú:

- Iteraciones del algoritmo: 100
- Permanencia tabú (t): 2
- Valor de penalización (P): 1000

Resultados

Se hicieron 100 corridas del algoritmo de búsqueda tabú. En el 100% de las corridas el algoritmo consigue la solución óptima, por consiguiente, el costo promedio es igual al costo óptimo 5457192,8 u.m. y el error relativo promedio es igual a 0%.

El tiempo promedio de ejecución del algoritmo de búsqueda tabú fue de 43,53 milisegundos, lo que corresponde a un 93,9% menos que el tiempo de ejecución del algoritmo exacto de ramificación y acotamiento en SAS, el cual fue de 710 milisegundos.

CAPÍTULO IV

Algoritmos Genéticos

4.1 Origen de los algoritmos genéticos

El término *"algoritmo genético"* (AG) fue usado por primera vez por John Holland (1975), al desarrollar una técnica de búsqueda inspirada en la teoría de la evolución de Darwin y en la genética. Esta técnica se basa en los mecanismos de selección que utiliza la naturaleza, de acuerdo a los cuales los individuos más aptos de una población son los que sobreviven, al adaptarse más fácilmente a los cambios que se producen en su entorno.

En el campo de la optimización, los algoritmos genéticos han sido utilizados con bastante éxito en una gran cantidad de problemas en diferentes áreas del conocimiento, para alcanzar soluciones óptimas o cercanas al óptimo cuando los métodos exactos presentan dificultades. Dentro de las metaheurísticas se han destacado como una técnica de optimización robusta y eficiente, y se consideran promisorios para solucionar problemas de optimización difíciles de resolver.

4.2 Principios del método

Los principios de la naturaleza en los cuales están inspirados los algoritmos genéticos son muy simples. De acuerdo con la teoría de Charles Darwin, el principio de selección privilegia a los individuos más aptos con mayor longevidad y, por lo tanto, con mayor probabilidad de reproducción. Por el contrario, individuos pocos dotados producirán un menor número de descendientes. Los individuos con más descendientes tienen más oportunidades de transmitir sus códigos genéticos a las próximas generaciones. Tales códigos genéticos constituyen la identidad de cada individuo y están representados en los cromosomas. La combinación de buenas características provenientes de diferentes ancestros, puede producir descendientes cuya adaptación es mucho mayor que la de cualquiera de sus ancestros. De esta manera, las especies evolucionan logrando características que les permiten adaptarse cada vez mejor al entorno en el que viven.

Los algoritmos genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución a un problema dado. A cada individuo se le asigna un valor que mide la calidad de esa solución, lo que en la terminología de los algoritmos genéticos se denomina grado de adaptación (*fitness*). En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos y poder sobrevivir. Cuanto mayor sea la adaptación de un individuo al problema o dicho de otra manera, cuanto mejor sea una solución, mayor será la probabilidad de que sea seleccionado para reproducirse, cruzando su material genético con otro individuo. Este cruce producirá nuevos individuos descendientes de los anteriores, los cuales comparten algunas de las características de sus padres. Además, los individuos descendientes que surgen del cruzamiento pueden mutar de acuerdo a cierta probabilidad, es decir, cambiar algunas de sus características. Los procesos de cruzamiento y mutación dan lugar a una nueva población, la cual reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior (Cabezas, 2002).

Este procedimiento se repite un determinado número de iteraciones (generaciones), después de varios ciclos de evolución la población deberá contener individuos más aptos y al final del proceso, si el algoritmo genético ha sido bien diseñado, la, población convergerá hacia una solución óptima del problema.

4.3 Conceptos básicos

Representación. Una de las características distintivas de los algoritmos genéticos es que utiliza una representación codificada del problema. Utilizando términos genéticos, se denomina *fenotipo* al conjunto de variables de decisión del problema y *genotipo* a la representación codificada de dichas variables de decisión (Reeves, 2003).

Un vector de variables de decisión **X** es representado por una cadena de caracteres *s* de longitud *l*. En muchas situaciones esta representación puede ser binaria, tal como en el problema que se ha venido tratando en los capítulos anteriores, en el cual el vector de variables de decisión es una secuencia de 0's y 1's y cada posición indica si un rodal es o no cortado en un año determinado. Cuando las variables de decisión del problema original son

binarias no hay diferencia entre el fenotipo y el genotipo, ya que la representación original puede ser utilizada en la implementación del algoritmo genético.

Hay otros casos en los que una representación discreta con una cardinalidad mayor que 2 puede ser apropiada, como por ejemplo un problema de asignación. Supóngase que se tienen 4 plantas de producción que deben ser ubicadas en 4 posibles sitios, una representación pudiera ser una cadena que guarde en la primera posición cuál planta irá en la localización 1, en la segunda posición cuál planta se ubicará en la localización 2 y así sucesivamente. Por ejemplo, la cadena 3241 indica que en la primera localización estará la planta 3, en la segunda localización la planta 2, en la tercera localización la planta 4 y en la cuarta localización la planta 1. Este es un problema de permutaciones y como éste hay muchos dentro del contexto de problemas de optimización difíciles de resolver.

También existe otro tipo de problemas en los que las variables de decisión son enteras (no binarias) o reales, en estos casos se requiere codificar en forma binaria las soluciones. Un ejemplo es el problema utilizado en el capítulo anterior para ilustrar algunos conceptos de la búsqueda tabú, en el que se desea maximizar una función $f(x) = x^3 - 60\,x^2 + 900\,x + 100$ y la variable x es codificada como un entero binario de 5 bits en el rango [0, 31], después de encontrar la solución óptima, ésta debe ser decodificada para obtener el valor de la variable de decisión original. Aunque la aplicación de técnicas heurísticas por lo general está dirigida a optimización discreta, es conveniente mencionar este tipo de problemas.

Función de adaptación. Esta función mide el grado de adaptación de un individuo, asociándole un valor único; este valor es un número real no negativo que es más grande cuanto mejor sea la solución propuesta por dicho individuo. Por consiguiente, la función de adaptación permite determinar cuáles individuos corresponden a buenas propuestas de solución y cuáles no.

La función de adaptación está relacionada con la función objetivo que se desea optimizar. Si el problema es de maximización, la adaptación debe indicar qué tan alto es el valor de la función objetivo para un individuo particular, en algunos casos pueden coincidir la función objetivo y la función de adaptación, sin embargo, este no es el caso más común, depende del problema particular y de la representación codificada que se utilice. Cuando se trata de un

problema de minimización, la adaptación para un individuo debe ser tanto más alta cuanto más pequeño sea su valor objetivo, por tal razón, es necesario establecer una equivalencia entre la función de adaptación y la función objetivo.

En muchas situaciones es necesario construir cuidadosamente la función de adaptación, ya que la calidad de esta función puede influir de manera considerable en la eficiencia de un algoritmo genético (Dréo et al., 2006).

Cromosoma. En los algoritmos genéticos, un cromosoma es una estructura de datos que representa una de las posibles soluciones del espacio de búsqueda del problema. Los cromosomas son sometidos a un proceso de evolución que envuelve evaluación, selección, cruzamiento y mutación, los cuales serán explicados con más detalles en los siguientes apartados.

Gen. Corresponde a cada uno de los elementos que conforman un cromosoma.

Alelos. Son los valores que pueden tomar los genes de un cromosoma.

En la figura 4.1 se representan algunos de los conceptos básicos de los algoritmos genéticos.

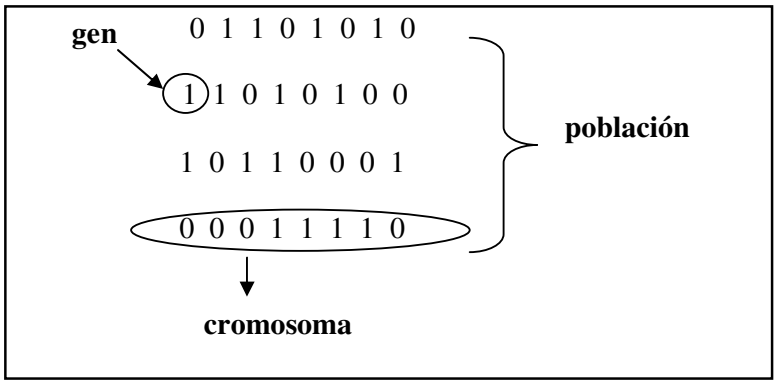


Figura 4.1 Conceptos básicos de los AG

4.4 Operadores genéticos

Los operadores genéticos son mecanismos que garantizan la evolución de los individuos, creando a partir de una población inicial nuevas poblaciones o generaciones de individuos mejorados. Los tres operadores básicos que operan en los algoritmos genéticos son selección, mutación y cruzamiento.

1) Selección

Este operador selecciona individuos en una población para su reproducción o supervivencia en el algoritmo genético. El objetivo fundamental del operador de selección es resaltar las mejores soluciones dentro de una población. Este operador no genera soluciones nuevas en el espacio de búsqueda, sino que determina qué individuos dejarán descendencia y en qué cantidad en la próxima generación De esta manera, la selección integra dos conceptos diferentes reproducción y selección. Cuando una o más copias de una buena solución son reproducidas, esta operación se llama reproducción. La selección se refiere al hecho de que múltiples copias de una solución son incluidas en la población y se eliminan algunas soluciones inferiores (Deb, 2000).

La identificación de buenas o malas soluciones está relacionada a la adaptación de una solución. La idea básica es que las soluciones que tienen mayor adaptación tienen mayor probabilidad de selección. Así, la selección permite orientar la búsqueda hacia aquellos puntos más promisorios, es decir, con la mayor adaptación observada hasta el momento. El operador de selección genera a partir de la población actual una población intermedia del mismo tamaño, reproduciendo con un mayor número de copias a los individuos más aptos y eliminando o asignando un menor número de copias a los individuos menos aptos (Estévez, 1997).

El proceso de selección puede ser descompuesto en dos fases:

1) Creación de una distribución de probabilidad basada en la adaptación.
2) Obtención de muestras de esta distribución.

Hay diferentes maneras de implementar el operador de selección, cada una de las cuales define en forma diferente la distribución de probabilidad y por ende, la forma en cómo asigna el número de copias de cada solución.

Selección proporcional. Este tipo de selección fue propuesta por Holland (1975) para los algoritmos genéticos genéricos. Se basa en la idea de que la probabilidad de selección P_i del i-ésimo individuo en la población depende de la adaptación relativa de éste con respecto a la población:

$$P_i = \frac{f_i}{\sum_{j=1}^{M} f_j} \qquad [4.1]$$

Donde M es la cantidad de individuos de la población y f_j es la adaptación del j-ésimo individuo.

Esta definición implica que la función de adaptación es positiva en el dominio de búsqueda y debe ser maximizada (a mayor valor de la función de adaptación, mayor es la probabilidad de selección). Para satisfacer estas condiciones puede ser necesario realizar algunas transformaciones simples a la función objetivo.

Proceso de muestreo

Método de la ruleta. Es el esquema básico para implementar la selección proporcional. Los pasos para aplicar este método son:

1. Relacionar uno a uno los individuos con segmentos contiguos de la recta real [0,1), tal que cada segmento individual sea proporcional a la probabilidad de adaptación.
2. Generar un número aleatorio en [0,1).
3. Seleccionar el individuo cuyo segmento cubre el número aleatorio.
4. Repetir el proceso hasta obtener el número deseado de individuos.

El método de la ruleta tiene una alta varianza, puede ocurrir que un individuo con una excelente adaptación no sea seleccionado y que un individuo con valores bajos de la función de adaptación sea seleccionado varias veces.

Muestreo estocástico universal. Este método de muestreo se emplea con bastante frecuencia debido a que tiene una varianza pequeña. Al igual que el método de la ruleta, el muestreo estocástico universal considera un segmento de la recta real [0,1) particionado en tantas zonas como individuos haya en la población y cada zona tiene un tamaño proporcional a la probabilidad de adaptación. La diferencia está en que en este procedimiento se genera un número aleatorio que ubica el primer individuo seleccionado y a partir de este seleccionan

los demás como en un muestreo estadístico sistemático. Es decir, si se tiene una población con M individuos habrán M punteros igualmente espaciados en la ruleta, cada uno de los cuales indicará un individuo que será seleccionado, de modo que en un solo lanzamiento se obtienen M individuos. Se diferencia del método de la ruleta en que este último sólo tiene un puntero que señala un individuo en una ruleta. Existen en la literatura algunos estudios que muestran la superioridad del muestreo estocástico universal, sin embargo en muchas aplicaciones se sigue utilizando el método de la ruleta por ser el propuesto originalmente por Holland (Reeves, 2003).

Ejemplo. Supóngase que se tiene una población de 10 individuos con los siguientes valores de la función de adaptación f_i y de la probabilidad de selección P_i.

individuo	1	2	3	4	5	6	7	8	9	10
f_i	231	144	364	300	364	175	204	399	175	204
P_i	0,09	0,06	0,14	0,12	0,14	0,07	0,08	0,16	0,07	0,08
P_i acumulada	0,09	0,15	0,29	0,41	0,55	0,62	0,70	0,85	0,92	1,00

Para aplicar el método de la ruleta se construye la representación de los individuos en la recta real [0,1), se obtiene un número aleatorio en el intervalo [0,1), y se elige el individuo cuya zona incluya el número aleatorio. En la figura 4.2 se muestra una ilustración de este método, en la cual se elige aleatoriamente el número 0.44 que corresponde al individuo 5.

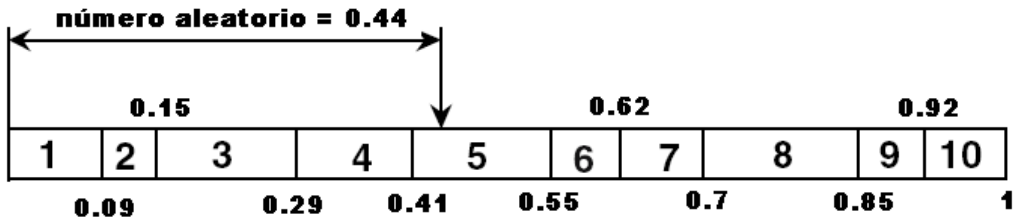


Figura 4.2. Representación del método de la ruleta

En la aplicación del muestreo estocástico universal se representan los individuos de la misma manera que en el método de la ruleta. Será necesario tener 10 punteros que indiquen

los individuos que serán seleccionados. La distancia entre punteros será igual a 1/*M*, en este caso 1/10 = 0.1. Se elige la ubicación del primer puntero, seleccionando un número aleatorio entre 0 y 0.1, a partir de esta ubicación sistemáticamente se obtienen los demás individuos. En la figura 4.3 se muestra una ilustración de este método, en la que se elige aleatoriamente el número 0.06 para la ubicación del primer puntero. Se observa en la figura que una vez que se ubican todos los punteros, el individuo 2 no es seleccionado, el 3 es seleccionado dos veces y el resto de los individuos es seleccionado una vez.

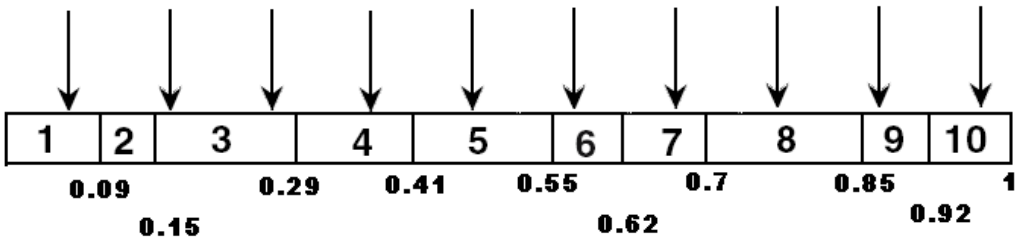


Figura 4.3. Representación del muestreo estocástico universal

Selección por Ordenamiento (Ranking). Este método ordena la población comenzando con el mejor individuo y finalizando en el peor, luego asigna una probabilidad a cada individuo proporcional a la posición que ocupa en el ordenamiento realizado. La primera posición es 0 y la última *M*-1.

Hay varias maneras de definir la probabilidad de selección mediante ordenamiento, una de ellas es el ordenamiento lineal. En el ordenamiento lineal es necesario definir un parámetro β que es el valor esperado de descendencia que tendrá el mejor individuo durante cada generación. La probabilidad de selección para el individuo *i* se define como:

$$P_i = \frac{\alpha + (posición(i)/(M-1))(\beta - \alpha)}{M} \qquad [4.2]$$

Donde α es el número de descendencia que tendrá el peor individuo y *M* es el tamaño de la población. Se puede demostrar que $\alpha = 2 - \beta$ y $1 \leq \beta \leq 2$. Los detalles de la demostración, así como también los detalles sobre el ordenamiento no lineal (probabilidades de selección están basadas en la posición del individuo en el ordenamiento pero no son proporcionales a ésta) pueden leerse en (Grefenstette, 2000).

Una vez calculadas las probabilidades de selección, la muestra de individuos para completar la población puede obtenerse mediante el muestreo estocástico universal, tal como se explico en la selección proporcional pero usando las probabilidad calculadas mediante [4.2].

Selección por torneo. En este método la selección se hace en base al número de premios que gana una solución en una competencia. Para ello, cada solución compite con algún número *q* de soluciones seleccionadas aleatoriamente de la población. La competencia se fundamenta en determinar si la solución que se está evaluando es tan buena o mejor que la solución aleatoria, en ese caso gana un premio, el proceso se repite para las *q* soluciones elegidas al azar. Así, una solución puede ganar hasta *q* premios. Esta competencia es conducida para toda las soluciones de la población y el método de selección por torneo elige un número de soluciones definido basado en el número de premios que ha ganado cada una de ellas. Para propósitos prácticos $q \geq 10$ es a veces considerada una selección dura, y q en el rango entre 3 y 5 es considerada una selección suave. La selección suave pudiera conducir a que el algoritmo se quede atrapado en óptimo local por un período de tiempo (Fogel, 2000).

2) Cruzamiento

Es una de las operaciones fundamentales que intervienen en todo algoritmo genético. Se aplica a dos individuos (padres) y consiste en una combinación de los mismos para obtener como resultado otros dos nuevos individuos (hijos). El cruzamiento se hace reemplazando algunos genes de un padre por los genes correspondientes del otro padre.

Una manera sencilla de implementar este operador es definir un punto de cruzamiento e intercambiar los genes de los padres siguientes a esta posición para dar origen a un hijo, luego intercambiar los genes anteriores (incluyendo el punto de cruzamiento) para crear otro hijo.

<u>*Ejemplo.*</u> Se tienen las s soluciones padre P1 y P2, se utiliza un punto de cruzamiento igual a 3 para obtener los hijos H1 y H2.

P1	0 1 1 0 1 0 1	H1	0 1 1 **1 0 1 1**
P2	1 0 1 1 0 1 1	H2	**1 0 1** 0 1 0 1

También puede hacerse el cruzamiento utilizando dos puntos de cruce. Para el ejemplo anterior se puede elegir dos números aleatorios entre 1 y 6 (cada padre tiene 7 genes) para hacer el cruzamiento. Supóngase que se eligen los números 2 y 4. Los hijos se obtienen al intercambiar en cada padre los genes que están a partir de la posición 2 hasta la posición 4, es decir, se intercambian los genes de las posiciones 3 y 4.

H1 0 1 **1 1** 1 0 1
H2 1 0 **1 0** 0 1 1

Una prescripción similar puede hacerse para *m* puntos de cruzamiento siendo $m > 1$.

Otro enfoque utilizado en el cruzamiento es el llamado cruzamiento uniforme, que consiste en aplicar este operador genético en forma completamente aleatoria. Para ello, se establece el porcentaje de genes que serán cruzados, se eligen en forma aleatoria la posición de los genes a intercambiar y se efectúa el cruzamiento de los genes seleccionados en ambos padres (Cabezas, 2002).

3) Mutación

El operador genético de mutación consiste cambiar alguna(s) de las características de alguno de los individuos para crear un nuevo individuo. La mutación escoge aleatoriamente un subconjunto de genes de un individuo y cambia su valor. Cuando la representación es binaria, la mutación simplemente cambia el valor de los genes seleccionados por el bit complementario.

Ejemplo. Se tiene la siguiente solución (individuo) para un problema binario:

0 1 1 1 0 1 0

Una mutación en el tercer y quinto gen origina una nueva solución:

0 1 **0** 1 **1** 1 0

El tamaño del subconjunto de genes puede ser un número predeterminado; así por ejemplo, cuando se cambia sólo un gen se le denomina al proceso mutación simple. Otra manera de implementar la mutación es examinar cada gen de un individuo y de acuerdo a una cierta

probabilidad (usualmente muy pequeña) cambiar su valor, en este caso no se sabe de antemano cuántos genes serán cambiados en la mutación (Reeves, 1996).

La mutación permite generar nuevas soluciones que exploren regiones del dominio del problema que probablemente no se han visitado aún. En tal sentido, la mutación favorece la diversidad en la población de soluciones.

4.5 Algoritmo Genético Simple

En su trabajo, Holland (1975) propone una manera de seleccionar individuos y cruzarlos. Actualmente existen otras propuestas, pero las de Holland constituyen la base de muchos desarrollos teóricos y prácticos en el área. Goldberg (1989) retomó el algoritmo de Holland y lo popularizó llamándolo *algoritmo genético simple* (AGS).

En el algoritmo genético simple se utiliza una representación binaria y se aplican los tres operadores genéticos básicos selección, cruzamiento y mutación. En detalle, los pasos que sigue el AGS se describen a continuación.

1. Comenzar con una población generada en forma aleatoria y compuesta por N cromosomas (soluciones candidatas al problema). A esta se le llamará población actual.
2. Evaluar cada solución de la población actual usando la función de adaptación.
3. Repetir los siguientes pasos hasta crear una descendencia de N individuos
    - Seleccionar un par de individuos (soluciones) padres de la población actual, de acuerdo a una probabilidad de selección que depende de la adaptación. El AGS utiliza el método de la ruleta pero pudiera utilizarse cualquier otro método de selección.
    - Con probabilidad P_c (probabilidad de cruzamiento), cruzar el par dos nuevos individuos. Si el cruzamiento no se lleva a cabo generar dos nuevos individuos que serán copias exactas de sus padres.
    - Mutar los dos nuevos individuos en cada gen con probabilidad P_m y colocar las dos soluciones resultantes en la nueva población.

- Si *N* es impar un individuo de la nueva población puede ser descartado aleatoriamente.

4. Reemplazar la población actual con la nueva población.
5. Repetir desde el paso 2 hasta el 4 un número determinado de iteraciones o hasta que se cumpla un condición de terminación.

Cada iteración de este proceso es llamada generación. Un algoritmo genético es típicamente iterado entre 50 y 500 generaciones o más. Al final de cada corrida hay uno o más cromosomas con una alta adaptación, aquel que tenga la mayor adaptación se considera la solución del problema. Ya que la aleatoridad juega un papel importante en cada corrida de un algoritmo genético, dos corridas con semillas aleatorias diferentes pueden presentar comportamientos diferentes. Es usual que en las investigaciones en las que se usan algoritmos genéticos se presenten estadísticas relacionadas con las soluciones con mejor adaptación y la iteración en que fueron alcanzadas (Mitchell, 1998).

En la figura 4.4 se presenta un diagrama de flujo con los pasos de un algoritmo genético simple.

El procedimiento descrito anteriormente es la base para diversas aplicaciones de algoritmos genéticos. Hay también versiones mucho más complicadas que utilizan representaciones distintas a las binarias o que utilizan formas diferentes de los operadores genéticos básicos de selección, cruzamiento y mutación, incluso pueden incluir otros operadores.

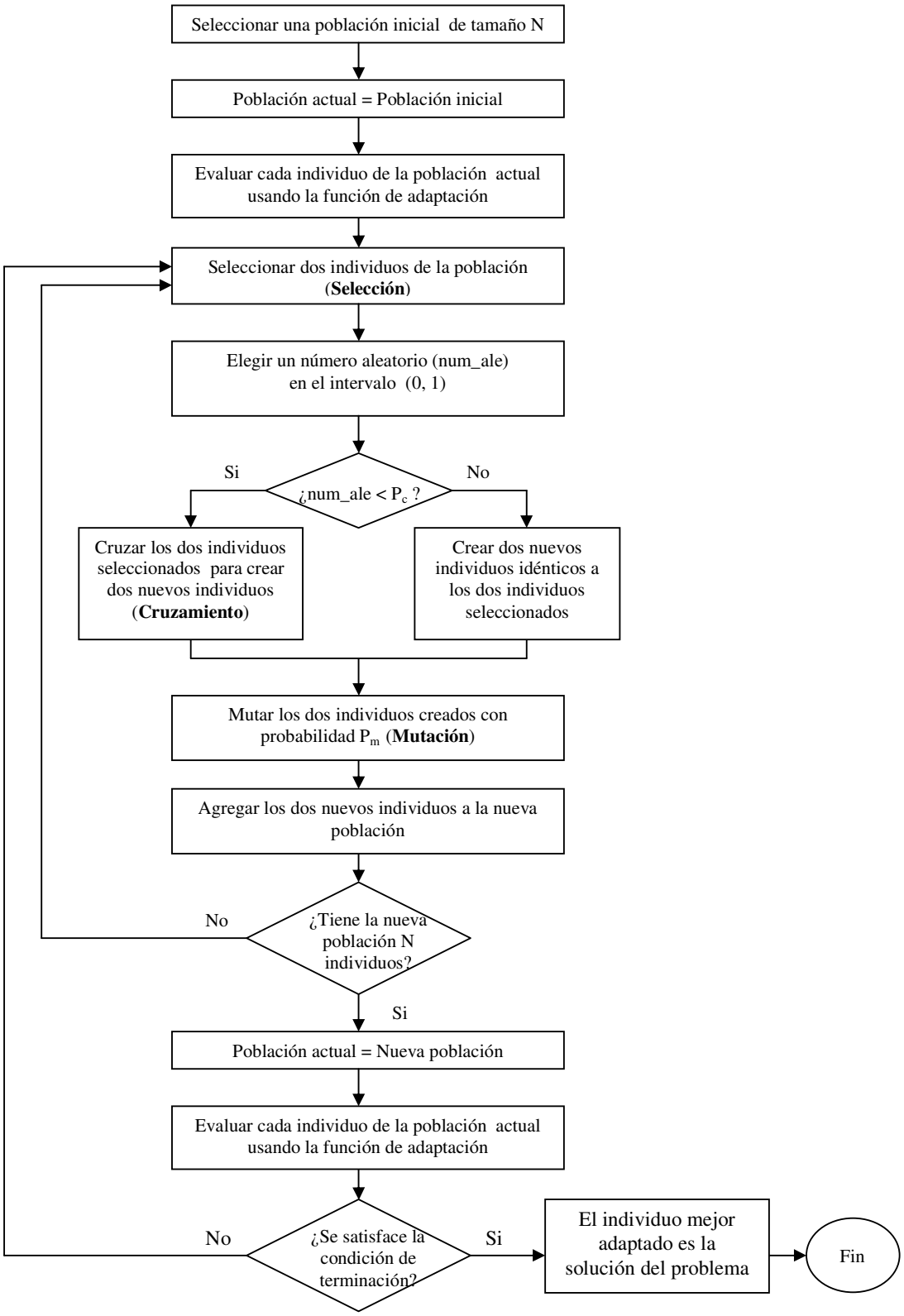


Figura 4.4. Pasos del algoritmo Genético Simple

Ejemplo sencillo. Con el objetivo de clarificar la mecánica de funcionamiento de un algoritmo genético se considera el siguiente problema (adaptado de Reeves, 1996):

$$\text{Maximizar } f(x) = x^3 - 60\,x^2 + 900\,x + 100$$

donde x es codificada como un entero binario de 5 bits en el rango [0, 31]. Es el mismo problema que se utilizó para ilustrar los conceptos básicos de búsqueda tabú en el capítulo 3.

Al comienzo del algoritmo se genera aleatoriamente la población inicial, la cual está conformada por cinco individuos o soluciones. En la siguiente tabla se encuentran los valores asociados a cada solución aleatoria

Tabla 4.1. Población inicial

No. de solución	Solución	x	f(x)	P_i (Probabilidad de selección)	P_i acumulada	Intervalos para cada solución
1	1 1 0 0 0	24	964	0,08	0,08	< 0.08
2	0 1 1 0 0	12	3988	0,34	0,43	[0.08, 0.43)
3	0 0 1 0 0	4	2804	0,24	0,67	[0.43, 0.67)
4	1 1 1 0 1	29	129	0,01	0,68	[0.67, 0.68)
5	0 1 1 1 0	14	3684	0,32	1,00	[0.68, 1)
		Sumatoria	11596	1		

Para seleccionar los padres que serán cruzados se procede a utilizar el método de la ruleta. En cada paso se obtienen dos números aleatorios en el intervalo [0,1) y en base a los intervalos definidos por la probabilidad de selección acumulada se determina a qué solución corresponde cada número aleatorio. Por ejemplo, los dos primeros números aleatorios obtenidos son 0,72 y 0,01, que corresponden a las soluciones 5 y 1, esto es, 0 1 1 1 0 y 1 1 0 0 0.

De cada par de padres se obtienen dos individuos mediante el cruzamiento. En este ejemplo se utiliza una probabilidad de cruzamiento igual a 1 (P_c = 1). El cruzamiento se hace seleccionando al azar el punto de cruce, el cual puede estar en las posiciones 1, 2, 3 ó 4. Supóngase que en el paso 1 se selecciona al azar la posición 3, entonces los descendientes obtenidos a partir de los padres 5 y 1 son:

P5 0 1 1 1 0 H1 0 1 1 0 0

P1 1 1 0 0 0. H2 1 1 0 1 0

El siguiente paso es llevar a cabo la mutación, en este caso se realiza con una probabilidad de 0.02 para cada gen. Supóngase que en el caso de H1 sólo mutará el gen 4 y que en H1 no se producirá una mutación, los descendientes después de la mutación son:

H1 0 1 1 **1** 0

H2 1 1 0 1 0 (no muta)

En la tabla 4.2 se resume el proceso para seleccionar una nueva población de cinco individuos. Debido a que cada cruce da lugar a dos individuos nuevos se obtienen 6 individuos para la nueva población, por lo que se puede eliminar alguno en forma aleatoria o aquél que tenga menor adaptación.

Tabla 4.2 Obtención de una población nueva a partir de la población inicial

Paso	Padre 1	Padre 2	Punto de cruce	Hijo	¿Mutación?	Nuevo individuo	f(x)
1	5	1	3	0 1 1 0 0	NNNSN	0 1 1 1 0	3684
1	5	1	3	1 1 0 1 0	NNNNN	1 1 0 1 0	516
2	2	1	1	0 1 0 0 0	NNNNN	0 1 0 0 0	3972
2	2	1	1	1 1 1 0 0	NSNNN	1 0 1 0 0	2100
3	3	3	2	0 0 1 1 0	NNNNN	0 0 1 1 0	3556
3	5	5	2	0 1 1 0 0	NNNNN	0 1 1 0 0	3988

Con la población obtenida se repite todo el proceso: cálculo de las probabilidades de selección y un nuevo ciclo de selección, cruzamiento y mutación.

4.6 Elementos a considerar al implementar un algoritmo genético

Para resolver un problema mediante un algoritmo genético se deben definir los siguientes aspectos:

- Codificación o representación genética de las posibles soluciones (cadenas binarias en el caso del algoritmo genético simple).
- Forma de crear la población inicial (aleatoria en el algoritmo genético simple).
- Función de adaptación que permitirá clasificar a los individuos.

- Detalles para la implementación de los operadores genéticos de reproducción, cruzamiento y mutación que simulan la evolución de las poblaciones.
- Valores de los parámetros del algoritmo: longitud de los individuos, tamaño de la población, número de generaciones, probabilidad de cruzamiento y probabilidad de mutación.

4.8 Problemas de optimización combinatoria resueltos con algoritmos genéticos

Los algoritmos genéticos han sido utilizados en múltiples aplicaciones en diversas áreas. Beasley (2000) señala que las aplicaciones pueden ubicarse por lo menos en una de las siguientes categorías.

- Planificación: dentro de esta categoría se encuentran los problemas de establecimiento de rutas, programación y empaque. En el establecimiento de rutas se ubica el clásico problema del vendedor viajero, problemas de transporte y diseño de rutas para el entrenamiento de robots. Los algoritmos genéticos también se utilizan para resolver problemas de difícil solución relacionados a la programación de actividades y horarios. Igualmente, los algoritmos genéticos han resuelto problemas de empaque en los que deben ubicarse un conjunto de ítems de tal forma que se minimice el espacio utilizado (ejemplo: ubicación de circuitos, cortes de un material para obtener varias piezas, ubicación de ítems en un espacio tridimensional).
- Diseño: los algoritmos genéticos han sido usados para diseñar sistemas electrónicos, sistemas digitales, circuitos integrados, topologías de redes neuronales, redes de telecomunicaciones, estructuras, aeronaves, aceleradores lineales, reactores químicos, entre otras aplicaciones.
- Simulación: lo algoritmos genéricos se han utilizado en problemas difíciles de simulación en el área de química y biología, como por ejemplo en la determinación del equilibrio de sistemas que reaccionan químicamente, en la determinación de la entalpía libre de los componentes de un sistema químico, determinación de la estructura tridimensional de una proteína, etc. También tienen aplicaciones en el desarrollo de modelos de evolución biológica, para simular como el sistema nervioso

aprende, entre otros. En el campo de la economía se han aplicado algoritmos genéticos como apoyo en la simulación de sistemas económicos.

- Control: se utilizan algoritmos genéticos en el diseño y construcción de controladores para diferentes tipos de sistemas.
- Clasificación: son muchas las aplicaciones de los algoritmos genéticos en los desarrollos teóricos y prácticos de los sistemas clasificadores, los cuales se utilizan para identificar y categorizar características de un fenómeno o sistema, para posteriormente tomar decisiones. Tienen aplicaciones en la robótica, control, lingüística, finanzas, biología, procesamiento de imágenes, juegos, entre otros.

Se espera que en el futuro las áreas de aplicación de los algoritmos genéticos sigan creciendo. En los últimos años una gran parte de la investigación se ha concentrado en el desarrollo de mejoras al desempeño de los algoritmos genéticos, se han propuesto nuevas técnicas de representación, selección y cruzamiento, con resultados muy alentadores que permitirán que se utilicen con éxito en otras áreas del conocimiento.

4.9 Aplicaciones en planificación forestal

Se han realizado algunas aplicaciones de los algoritmos genéticos para resolver problemas complejos de planificación forestal, mostrando resultados promisorios tanto en la planificación espacial como en la no espacial. Dentro de los primeros trabajos se encuentra el modelo de programación de cosechas desarrollado por Falcão y Borges (2001), el cual utiliza variables binarias para representar las alternativas de manejo de cada rodal con restricciones de volumen y aplica como técnica de solución un algoritmo genético. Ducheyne et al. (2004) utilizan un algoritmo genético para optimizar un objetivo y un algoritmo genético multiobjetivo para resolver un problema de programación de cosechas en el que se busca maximizar el valor presente neto y minimizar las desviaciones entre períodos de corta sucesivos, y hacen comparaciones entre ambos algoritmos. Rodrigues et al. (2004) desarrollaron un algoritmo genético para resolver cuatro problemas de planificación forestal en lo que se desea maximizar el valor presente neto con restricciones de singularidad, producción máxima y mínima. Las investigaciones antes mencionadas no incluyen consideraciones espaciales.

Dentro de los modelos que tienen restricciones espaciales está el de Lu et al. (2000), quienes desarrollaron un algoritmo genético para delimitar unidades de manejo forestal en el marco de la planificación operacional. También los modelos de Ducheyne et al. (2006) y Thompson et al. (2009) implementan diferentes estrategias en los algoritmos genéticos para resolver problemas con restricciones espaciales.

En la literatura hay algunos artículos en los que se hacen comparaciones entre distintas heurísticas para abordar problemas de planificación forestal de diferente índole (espacial y no espacial), principalmente se analizan las técnicas de recocido simulado, búsqueda tabú, algoritmos genéticos y otros métodos de búsqueda escaladores (*hill-climbing*). En tal sentido, pueden mencionarse los trabajos de Falcão y Borges (2002), Palahí et al. (2004), Pukkala y Kurttila (2005), Liu et al. (2006).

4.10 Ejemplo de aplicación

Se resolvió el mismo problema tratado en los capítulos II y III usando un algoritmo genético programado en Visual Basic 6.0. La implementación se realizó de acuerdo a las siguientes estrategias:

a. Representación de una solución: como las variables son binarias no es necesario usar otra representación del problema, se trabaja con las variables originales X_{ij} (se corta o no el rodal *i* el año *j*) y se mantiene la representación vectorial utilizada en los algoritmos anteriores.

b. Generación de la población inicial: se seleccionaron aleatoriamente *n* individuos o soluciones para conformar la población inicial. La generación de cada individuo se hizo siguiendo el mismo procedimiento utilizado para generar una solución inicial en los algoritmos de recocido simulado y búsqueda tabú, explicados en el ejemplo de aplicación de los capítulos II y III.

c. Función de adaptación (fitness): es necesario definir una función de adaptación que mida la bondad de una solución, por definición esta función debe ser positiva y sus valores deben ser mayores cuanto mejor sea la solución. Como se trata de un problema de minimización, no puede usarse la función objetivo como función de adaptación, por lo tanto es necesario hacer alguna transformación.

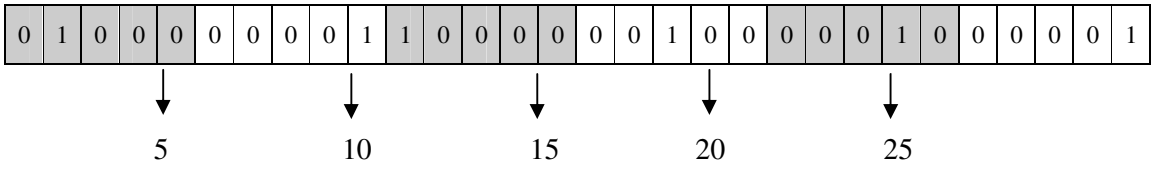
La función de adaptación utilizada es la siguiente:

$$\text{Fit}(\mathbf{X}) = \text{cota} - f^p(\mathbf{X}) \qquad [4.3]$$

Donde $f^p(\mathbf{X})$ es la función de costo penalizada definida en la ecuación 2.11 y utilizada tanto en recocido simulado como en búsqueda tabú, y la cota corresponde al mayor valor que puede tomar la función objetivo penalizada, el cual es 86512170. En este caso se conoce el mayor valor que puede alcanzar $f^p(\mathbf{X})$; como es un problema pequeño diseñado con fines didácticos, fue posible hacer un listado de todas la posibles soluciones . En problemas reales se utiliza como cota un valor alto que no pueda ser alcanzado por la función objetivo penalizada.

c. Selección: se utilizó un procedimiento de selección proporcional al valor de adaptación y el método de la ruleta.

d. Cruzamiento: se aplicó cruzamiento de un punto. Para preservar la estructura original de una solución y las restricciones de singularidad, los puntos de cruzamiento que pueden utilizarse están restringidos a aquellas posiciones del vector donde finaliza un rodal (5, 10, 15, 20, 25), entre estos valores se selecciona aleatoriamente el punto de cruzamiento.



*e. Mutación***:** se determina de acuerdo a la probabilidad de mutación si habrá o no mutación. En caso de que haya mutación, se elige aleatoriamente el rodal que mutará y dentro de este rodal el gen que mutará. Si el gen a mutar tiene un valor 0 se le asigna 1 y al resto de los genes de ese rodal se le asigna 0. Si el gen a mutar tiene un valor de 1 se le asigna 0 y se elige al azar entre los genes restantes uno al cual se le asignará el valor 1.

f. Parámetros del algoritmo genético: se definieron mediante experimentación.

- Tamaño de la población: 100
- Iteraciones del algoritmo: 100
- Probabilidad de cruzamiento: 1

- Probabilidad de mutación: 0,05
- Valor de penalización : 1000

Resultados

Se realizaron 100 corridas del algoritmo genético. En el 47% de las corridas el algoritmo encontró la solución óptima y en el restante 53% de los casos obtuvo soluciones factibles cercanas al óptimo. El costo promedio obtenido fue 5470376,39 u.m.

El error relativo promedio para las 100 corridas fue de 0,24%.

El tiempo promedio de ejecución del algoritmo genético fue 585,23 milisegundos.

CONCLUSIONES

Los métodos heurísticos y de manera especial las llamadas metaheurísticas tienen un enorme potencial en la resolución de problemas de planificación forestal, dada la complejidad implícita es este tipo de problemas. Las metaheurísticas estudiadas en esta monografía han mostrado su efectividad en diferentes aplicaciones citadas en la literatura, desde problemas de planificación sencillos hasta problemas con restricciones complejas tales como restricciones espaciales o no lineales. Asimismo, pueden encontrar buenas soluciones a problemas de grandes dimensiones en un tiempo de computación aceptable.

No es posible definir cuál de los métodos heurísticos es mejor, ya que su desempeño depende del problema particular que se esté tratando, en algunas aplicaciones puede resultar más eficiente alguna de las técnicas, pero esos resultados pueden variar para otros casos. No obstante, al momento de seleccionar alguna de las metaheurísticas aquí tratadas (recocido simulado, búsqueda tabú y algoritmos genéticos) para resolver un problema, vale la pena conocer cuáles son las ventajas y las desventajas que cada una de ellas puede presentar.

El recocido simulado es una técnica que generalmente logra una buena calidad de la solución, es decir el mínimo absoluto o un buen mínimo relativo, además es un método general y fácil de implementar para resolver problemas de optimización de diferente índole. Sin embargo, requiere de muchos parámetros para los cuales es necesario hacer ajustes empíricos y los resultados son muy sensibles a los valores de los parámetros.

La búsqueda tabú utiliza procedimientos de memoria adaptativa (se actualiza en función del tiempo y del análisis del entorno) que le permiten hacer una búsqueda en el espacio de soluciones en forma efectiva y económica, pudiendo entrar en complejidades que usualmente son difíciles de abordar con otros enfoques alternativos. Es técnica robusta ha mostrado muy buen desempeño en una amplia variedad de problemas, y en muchos casos encuentra soluciones superiores a las obtenidas por otros métodos. El principal inconveniente es que existen muchas variantes de la búsqueda tabú, por lo que en algunos casos puede requerirse de un estudio cuidadoso para determinar cuáles son las estrategias más adecuadas para un problema determinado.

En cuanto a los algoritmos genéticos, éstos se han constituido como una metaheurística robusta con muchos campos de aplicación que van más allá de la solución de problemas de optimización. Una de sus ventajas es que resultan menos afectados por los óptimos locales que otras técnicas, y además, son fáciles de entender y manipular. Su principal inconveniente es que en algunos casos pueden tardar mucho en converger, o no converger en absoluto.

Las técnicas de recocido simulado, búsqueda tabú y algoritmos genéticos dieron buenos resultados para el ejemplo de aplicación utilizado en esta monografía. En términos de eficiencia y rapidez resultó mejor la búsqueda tabú, seguida por el algoritmo de recocido simulado y el algoritmo genético. La búsqueda tabú en todas las corridas encontró la solución óptima, el recocido simulado la encontró en 87% de los casos y el algoritmo genético 47% de las veces. El hecho de que las dos últimas técnicas hayan encontrado soluciones factibles diferentes a la solución óptima en un cierto porcentaje de corridas, no debe ser visto como poco eficiente, está claro que no siempre los algoritmos heurísticos encuentran la solución óptima, pero sí se espera que encuentren buenas soluciones. En los resultados obtenidos se puede observar que el error relativo promedio de estas técnicas fue de 0,04% para la búsqueda tabú y 0,24% para el algoritmo genético, estos valores pequeños para el error indican que las soluciones encontradas diferentes a la solución óptima son buenas soluciones cercanas al óptimo. Obtener diferentes soluciones cercanas al óptimo puede ser una ventaja para el planificador, ya que tiene varias alternativas para seleccionar aquella que sea más conveniente de implementar en la práctica; en ocasiones tener únicamente la solución óptima puede ser problemático a la hora de llevar la solución a la realidad.

El tiempo de ejecución promedio de los algoritmos heurísticos utilizados fue menor en todos los casos que el tiempo de ejecución del algoritmo de ramificación y acotamiento. Aunque en este problema las diferencias en el tiempo de ejecución de los algoritmos son imperceptibles a simple vista, ya que están en el orden de los milisegundos, en problemas de mayor envergadura el tiempo de computación necesario para conseguir una solución es un factor muy importante.

Esta monografía es sólo una introducción al uso de técnicas heurísticas en la planificación forestal. En la literatura especializada los métodos heurísticos siguen encontrando

aplicaciones debido a la efectividad que han mostrado al resolver problemas complejos en el ámbito forestal. Esta es un área de investigación que se encuentra activa y que se espera continúe generando buenos resultados.

REFERENCIAS BIBLIOGRÁFICAS

Aruga K., J. Sessions, A.E. Akay. 2005. Heuristic planning techniques applied to forest road profiles. Journal of Forest Research 10(2): 83-92.

Baskent E.Z. 2001. Combinatorial Optimization in Forest Ecosystem Management Modeling. Turkish Journal of Agriculture and Forestry 25: 187 – 194.

Beasley D. 2000. Possible applications of evolutionary computation. En: Bäck T, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation I - Basic Algorithms and Operators, pp 4 - 19. Institute of Physics Publishing. Bristol, Reino Unido.

Bettinger P., K. Boston, J.P. Siry y D.L. Grebner. 2009. Forest Management and Planning. Elseiver, Academic Press. 331 p.

Bettinger P., K. Boston, Y.H. Kim y J. Zhua. 2007. Landscape-level optimization using tabu search and stand density-related forest management prescriptions. European Journal of Operational Research 176(2): 1265-1282.

Bettinger P., J. Sessions y K. Boston. 1997. Using Tabu search to schedule timber harvests subject to spatial wildlife goals for big game. Ecological Modelling 94(2-3): 111-123.

Boston K. y P. Bettinger P. 1999. An analysis of Monte Carlo integer programming, simulated annealing, and tabu search heuristics for solving spatial harvest scheduling problems. Forest Science 45 (2): 292 – 301.

Brumelle S., D. Granot, M. Helme e I. Vertinsky. 1998. A tabu search algorithm for finding good forest harvest schedules satisfying green-up constraints. European Journal of Operational Research 106(2-3): 408-424.

Cabezas C.A. 2002. Algoritmos genéticos, una opción para la optimización de funciones. Tecnología química 23(2): 65 – 69.

Cerny, V. 1985. A thermodynamical approach to the travelling salesman problem: an efficient simulation algorithm. Journal of Optimization Theory and Applications 45: 41 – 55.

Chen B. W. y K. Von Gadow. 2002. Timber harvest planning with spatial objectives, using the method of simulated annealing. Forstwissenschaftliches Centralblatt 121(1): 25-34.

Chen, B.W. y K. Von Gadow. 2008. Optimisation of spatial objectives in planning for sustainable forest medium-term management of Norway Spruce from northern Germany. Forest Research 21(3): 279-288.

Chiari R., O. Carrero, M. Jerez, M.A. Quintero y J. Stock. 2008. Modelo preliminar para la planficación del aprovechamiento en plantaciones forestales industriales en Venezuela. Interiencia 33 (011): 802 – 809.

Chung W.A. y J. Sessions. 2003. An application of combinatorial optimization heuristics to timber sale scheduling considering distance dependent costs and equipment balancing requirements. Managing forest Ecosystems 7 (91): 91 -103.

Crowe K. A. y J. D. Nelson. 2005. An evaluation of the simulated annealing algorithm for solving the area-restricted harvest-scheduling model against optimal benchmarks. Canadian Journal of Forest Research-Revue Canadienne De Recherche Forestiere 35(10): 2500-2509.

Dahlin B. y O. Sallnas. 1993. Harvest scheduling under adjacency constraints - A case study from the Swedish sub – alpine region. Scandinavian Journal of Forest Research 8: 281 – 290.

Deb K. 2000. Introduction to selection. En: Bäck T, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation I - Basic Algorithms and Operators, pp 166-171. Institute of Physics Publishing. Bristol, Reino Unido.

Díaz A., J.A. Ferland, C.C. Ribeiro, J.R. Vera y A. Weintraub. 2007. A tabu search approach for solving a difficult forest harvesting machine location problem. European Journal of Operational Research 179(3): 788-805.

Dowsland K. A. y B. A. Díaz. 2001. Diseño de Heurísticas y Fundamentos del Recocido Simulado. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 20: 34 – 52.

Dréo J., A. Pétrowski, P. Siarry y E. Taillard. 2006. Metaheuristics for hard Optimization. Springer - Verlag. Berlín. 369 p.

Duarte A., J. J. Pantrigo y M. Gallego . 2008. Metaheuristicas. Editorial Dykinson. Madrid. 244 p.

Ducheyne E.I., R.R. De Wulf y B. De Baets. 2004. Single versus multiple objective genetic algorithms for solving the even-flow forest management problem. Forest Ecolgy and Management 201(2-3): 259-273.

Ducheyne E.I., R.R. De Wulf y B. De Baets. 2006. A spatial approach to forest – management optimization_ linfing GIS and multiple objective genetic algorithms. International Journal of Geographical Information Science 20(8):917 – 928.

Estévez P. 1997. Optimización mediante algoritmos genéticos. Analaes del Instituto de Ingenieros de Chile, Agosto 97: 83 – 92.

Falcão A.O. y J.G. Borges. 2001. Designing an evolution program for solving integer forest management scheduling models: an application in Portugal. Forest Science 47(2): 158 – 168.

Falcão A.O. y J.G. Borges. 2002. Combining random and systematic search heuristic procedures for solving spatially constrained forest management scheduling models. Forest Science 48(3):608-621.

Fogel D.B. 2000. Other selection methods. En: Bäck T, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation I - Basic Algorithms and Operators, pp 201-204. Institute of Physics Publishing. Bristol, Reino Unido.

Gendreau M. 2003. An introduction to tabu search. En: Glover F y GA Kochenberger (Eds.). Handbook of metaheuristics, pp. 37-54. Kluwer Academic Publishers. Boston.

Glover F. 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13: 533–549.

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

Glover F. y B. Melián. 2003. Búsqueda Tabú. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19: 29-48.

Glover F. y M. Laguna. 1997. Tabu Search. Kluwer Academic Publishers. Norwell, Massachusetts. 382 p.

Glover F. y M. Laguna. 2002. Tabu Seach. En: Pardalos PM, y MGC Resende (Eds.), Handbook of Applied Optimization, Oxford University Press, pp. 194-208.

Gökçe M. A. 2007. Simulated Annealing (AI notes). Department of Industrial Systems Engineering. Faculty of Computer Sciences. Izmir University of Economics. Izmir, Turquía. [Publicación en línea]. Disponible desde Internet en: <http://homes.ieu.edu.tr/~agokce/Courses/simulatedannealing_ai_notes.doc> [con acceso el 27-02-09].

Goldberg D. 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley. Reading, EEUU.

Grefenstette J. 2000. Rank-based selection. En: Bäck T, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation I - Basic Algorithms and Operators, pp 187-194. Institute of Physics Publishing. Bristol, Reino Unido.

Hetz A., Taillard, D. de Werra. 1995. A Tutorial on Tabu Search. En: Proceedings of Giornate di Lavoro AIRO'95, Entreprise Systems: Management of echnological and Organizational Changes, pp. 13-24.

Heinonen T. y T. Pukkala. 2004. A comparison of one- and two-compartment neighbourhoods in heuristic search with spatial forest management goals. Silva Fennica 38(3): 319-332.

Holland J.H. 1975. Adaptation in natural and artificial systems. University of Michigan Press. Ann Harbor, Estados Unidos. 334 p.

Kirpatrick S., C. D. Gellat y M. P. Vecchi. 1983. Optimization by Simulating Annealing. Science 220 (4598): 671 – 679.

Laroze A.J. 1999. A linear programming, tabu search method for solving forest-level bucking optimization problems.Forest Science 45 (1): 108-116.

Laroze A.J y B.J. Greber. 1997. Using Tabu search to generate stand-level, rule-based bucking patterns. Forest Science 43 (2): 157-169.

Liu G., S. Han, X. Zhao, J.D. Nelson, H. Wang y W. Wang. 2005. Optimisation algorithms for spatially constrained forest planning. Ecological Modelling 194 (2006):421–428.

Liu G, S. Han, X. Zhao, J. D. Nelson, H. Wang y W. Wang. 2006. Optimisation algorithms for spatially constrained forest planning. Ecological Modelling 194(4): 421-428.

Lockwood, C. y T. Moore. 1993. Harvest scheduling with spatial constraints: a simulated annealing approach. Canadian Journal of Forest Research 23: 468 - 478.

Lu F. y L.O. Eriksson. 2000. Formation of harvest units with genetic algorithms. Ecolgy and Management 130(1-3): 57-67.

Lundy, M. y A. Mess. 1986. Convergence of an annealing algorithm. Mathematical Programming 34: 11 – 124.

Martí R. 2003. Procedimientos Metaheurísticos en Optimización Combinatoria. Matemàtiques 1 (1): 3-62.

Melián B., y F. Glover, 2006. Introducción a la búsqueda tabú. [documento en línea]. Disponible desde Internet en : <http://webpages.ull.es/users/mbmelian/TS_Spanish_GloverMelian(11-9-06).pdf > [con acceso el 29-04-09].

Melián B., J.A. Moreno, J.M. Moreno. 2003. Metaheuristicas: Una visión global. Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial 19 (2): 7-28.

Metropolis N., A.W. Rosenbluth, M.N. Rosenbluth., A.H. Teller y E. Teller. 1953. Equation of state calculation by fast computing machines. Journal of Chemistry 21: 1087 – 1091.

Mitchell M. 1998. An introduction to genetic algorithms. Massachusetts Institute of Technology Press. Cambridge, Massachussets, EEUU. 158 p.

Moins S. 2002. Implementation of a simulated annealing algorithm for Matlab. Training performed in Electronic Systems Linköping Institute of Technology. Linköping University Electronic Press. Linköping , Suecia. 34 p.

Moreno J.M. 1999. Heurísticas, apuntes de la asignatura Heurísticas. Universidad de la Laguna, España. [documento en línea]. Disponible desde Internet en : <webpages.ull.es/users/jmmoreno/Tema1-2.htm> [con acceso el 16-04-09].

Murray A. y R. Church. 1995. Heuristic solution approaches to operational forest planning problems. Operation Research Spektrum 17: 193 – 203.

Osman I.H. y J.P Kelly. 1996. Meta-heuristics: an overview. En: Osman I.H. y J.P Kelly (Eds.), Meta-Heuristics: Theory and Applications, pp 1 – 22. Kluwer Academic Publishers. Boston.

Palahí, M. y T. Pukkala. 2004. Métodos de optimización heurística para la resolución de modelos de planificación forestal.[documento en línea]. Disponible desde Internet en: <www.medforex.net/papers/forest/palahí-pukkala2.pdf> [con acceso el 13-04-09].

Palahí, M., T. Pukkala y A. Trasobares. 2004. Presentación del sistema de planificación forestal MONTE. .[documento en línea]. Disponible desde Internet en: <www.gruponahise.com/simposio/papers%20pdf/11%20Marc%20palahi.pdf.> [con acceso el 13-04-06].

Palahí, M., T. Pukkala, L. Pascual y A. Trasobares. 2004. Examining alternative landscape metrics in ecological forest planning: a case for capercaillie in Catalonia. Investigación Agraria: Sistemas y Recursos Forestales 13(3):527-538.

Pukkala, T. y M. Kurttila. 2005. Examining the performance of six heuristic search techniques in different forest planning problems. Silva Fennica 39(1): 67 – 80.

Reeves C.R. 1996. Modern Heuristic Techniques. En: Rayward-Smith V.J., I.H. Osman, C.R. Reeves y G.D. Smith (Eds.), Modern Heuristic Search Methods, pp 1- 25. Jon Wiley & Sons. Chichester, Inglaterra.

Reeves C.R. 2003. Genetic Algorithms. En: Glover F y GA Kochenberger (Eds.). Handbook of metaheuristics. Kluwer Academic Publishers, Boston, pp. 37-54

Richards E. W. y E. Gunn. 2000. A Model and tabu search method to optimize stand harvest and road construction schedules. Forest Science 46 (2): 188-203.

Richards E.W. y E.A. Gunn. 2003. Tabu search design for difficult forest management optimization problems. Canadian Journal of Forest Research 33(6): 1126 – 1133.

Rodrigues F. L., H.G. Leite, H.N. Santos y A.L de Souza. 2003. Solução de problemas de planejamiento florestal com restrições de inteireza utilizando Busca Tabu. Árvore 27(5): 701 - 713.

Rodrigues F. L., H.G. Leite, H.N. Santos, A.L de Souza y G.F. Silva. 2004. Metaheurística algoritmo genético para solução de problemas de planejamiento florestal com restrições de integridade. Árvore 28(2): 233 – 245.

Rodrigues F. L., H.G. Leite, H.N. Santos, A.L de Souza y G.F. Silva. 2004. Metaheurística simulated annealing para solução de problemas de planejamiento florestal com restrições de integridade. Árvore 28(2): 247 – 256.

Schwefel H.P. 2000. Advantages (and disadvantages) of evolutionary computation over other approaches. En: Bäck T, D.B. Fogel, Z. Michalewicz (Eds.), Evolutionary Computation I - Basic Algorithms and Operators, pp 20-22. Institute of Physics Publishing. Bristol, Reino Unido.

Tarp P. y F. Helles. 1997. Spatial optimization by simulated annealing and linear programming. Scandinavian Journal of Forest Research 12 : 390 – 402.

Thompson M-P, J.D. Hamann y J. Sessions. 2009. Selection and penalty strategies for genetic algorithms designed to solve spatial forest planning problems. International Journal of Forestry Research 2009: 1-14.

Troncoso J. J. 2002. Modelos de Planificación Forestal Orientados a la producción y Logística. Agronomía y Forestal UC (15): 15 – 18.

Van Deusen P.C. 1999. Multiple solution harvest scheduling. Silva Fennica 33: 207 – 216.

Voß S. 2001. Meta-heuristics: The State of the Art. En: A. Nareyek (Eds.), Local Search for Planning and Scheduling, LNAI 2148, pp 1 – 23. Springer – Verlag. Berlin.

Zanakis S. H. y J. R. Evans. 1981. Heuristic Optimization: why, when and how to use it. Interfaces 11(5):84–90.