Capítulo 5

Búsqueda Tabú

5.1 Introducción

Búsqueda Tabú (Glover, 86) es una estrategia para resolver problemas de optimización combinatoria. Algo muy parecido sugierió Hansen al mismo tiempo, y que llamó *steepest ascent/mildest descent*.

Muchas técnicas usan variantes de *local search*.

1. Seleccionar una $x \in X$ inicial.
2. Seleccionar algún $s \in N(x)$ (vecino) tal que: $c(s) < c(x)$.
3. *Si* no existe, x es un óptimo local y para.
4. *Sino*, sea $x := s$ y regresa al punto 2.

Limitante clave: se para al encontrar un mínimo local.

Búsqueda tabú combina búsqueda local con una heurística para evitar parar en mínimos locales y evitar entrar en ciclos.

La idea básica de búsqueda tabú es continuar con LS cuando se llega a un mínimo local al permitir movimientos que no mejoran la solución.

Para evitar regresar a soluciones pasadas y ciclarse, usa una memoria temporal, llamada lista tabú, que guarda la historia reciente de la búsqueda.

En resumen, búsqueda tabú es una combinación de búsqueda local con un mecanismo de memoria a corto plazo.

Elementos claves en búsqueda Tabú:

- Restricciones Tabú: restringir la búsqueda al clasificar ciertos movimientos como prohibidos (tabú), para evitar caer en soluciones recientemente generadas.
- Criterio de aspiración (*aspiration criteria*): liberar la búsqueda por medio de una función de memoria a corto plazo (olvido estratégico).

Se crea un subconjunto $T \subseteq S$ usando información histórica extendiendola t iteraciones en el pasado.

La membresía en T puede ser por medio de condiciones a cumplir (i.e., no necesariamente un índice de soluciones pasadas).

5.2 Algoritmo

El algoritmo se muestra en la tabla 5.1.

La figura 5.1 muestra un ejemplo sencillo del funcionamiento de búsqueda Tabú. Supongamos que en cada punto solo pueden hacerse dos movimientos, hacia un estado con un número anterior o a un estado con un número posterior y supongamos que nuestra lista tabú es de tamaño 3.

Supongamos que inicialmente estamos puestos en el punto marcado con el número 1 en la figura 5.1. De ahí se pueden hacer dos movimientos (hacia 0 y hacia 2). Se elige el mejor (hacia 2), denotemoslo como $mov(1, 2)$ y se actualiza la lista tabú (inicialmente vacía). Registramos el movimiento inverso en la lista ($mov(2, 1)$) para evitarlo en el futuro.

De ahí nos movemos hacia el estado 3 (el mejor y el permitido dada nuestra

Tabla 5.1: Algoritmo de Búsqueda Tabú.

1. Selecciona un estado $x \in X$ inicial y sea $x^* := x, k = 0$ (contador de iteración) y $T := \emptyset$.

2. *Si* $S(x) - T = \emptyset$ ve a 4,
   Sino, sea $k := k + 1$ y selecciona $s_k \in S(x) - T$ tal que $s_k(x) = OPTIMO(s(x) : s \in S(x) - T))$.

3. Sea $x := s_k(x)$. Si $c(x) < c(x^*)$ (donde x^* es la mejor solución encontrada hasta el momento), sea $x^* := x$.

4. *Si* se agotó el número de interacciones o si $S(x) - T = \emptyset$ *entonces* para, *sino*, actualiza T (añade el movimiento actual a la lista tabú y posiblemente elimina el elemento más viejo) y regresa a 2.

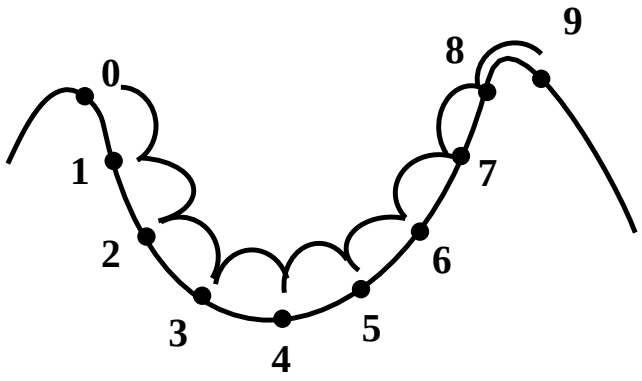

Figura 5.1: Ejemplo sencillo de búsqueda Tabú.

lista tabú actual) y después al 4, con lo que la lista tabú se llena con tres elementos: $\{mov(4,3), mov(3,2), mov(2,1)\}$.

El siguiente movimiento es peor en la función objetivo ($mov(4,5)$), pero es el mejor dentro de los permitidos por la lista tabú y por nuestro esquema de vecindad utilizado. También provoca que se elimine el elemento más viejo que tiene la lista tabú (por estar llena). La lista tabú queda como: $\{move(5,4), mov(4,3), mov(3,2)\}$.

Este proceso continua hasta que finalmente salimos del mínimo local al movernos del estado 8 al 9.

Puntos:

- Las soluciones dependen de como se actualiza T.
- No hay condición de óptimo local.
- Se busca la "mejor" solución en cada paso (función OPTIMO), en lugar de alguna opción que mejore la solución.

OPTIMO puede ser:

$$c(s_k(x)) = \text{ mínimo}(c(s(x)) : s \in S(x) - T)$$

OPTIMO da la mejor solución o la menos peor sujeta a la lista tabú.

En principio, se podría tomar alguna solución que mejore (en caso de que sea difícil encontrar la mejor), pero normalmente se sigue la estrategia más agresiva.

La razón principal es que los óptimos locales no presentan problemas.

Normalmente la lista tabú se implementa como una lista circular, añadiendo elementos en la posición 1 y eliminando los que sobrepasen la posición t (para una t fija).

Una forma efectiva de implementar la lista T sería:

$$T = \{s^{-1} : s = s_h \text{ para } h > k - t\}$$

donde k es el número de iteración y s^{-1} es el movimiento inverso de s.

Por lo que el paso de actualización de T sería: $T := T - s_{k-t}^{-1} + s_k^{-1}$.

En general, lo que se trata de evitar en caer en estados solución previos.

Esto no quiere decir que se tenga que escoger una t muy grande.

En la práctica T no toma la forma anterior: (i) s^{-1} previene un conjunto más grande de movidas, (ii) por consideraciones de memoria es deseable guardar sólo un subconjunto de atributos que caracterizan el movimiento.

Bajo estas condiciones, la lista tabú representa una colección de movidas C_h.

Cuando la lista $S - T = \emptyset$ se pueden eliminar los elementos más viejos de T para permitir continuar con el proceso.

5.3 Niveles de Aspiración

Con la lista tabú se evitan ciclos si se previene moverse de x a $s(x)$ si: (1) $s(x)$ yá ha sido visitado antes (muy caro en memoria e implementación); (2) el movimiento s yá se aplicó a x antes; (3) el movimiento s^{-1} yá se aplicó a $s(x)$ antes.

La lista tabú puede prohibir movimientos deseables que no produzcan ciclarse o también nos puede llevar a un punto en donde no es posible moverse.

Todos los algoritmos de búsqueda tabú permiten revocar o cancelar tabús.

A estos se les llama criterios de aspiración (*aspiration criteria*), que permiten movimientos, aunque sean tabú.

Lo más común es permitir movimientos que producen una mejor solución que la mejor solución actual, osea si se hace un movimiento que va de una x a $s(x)$ si $c(s(x)) < MEJOR(c(x))$.

Si se piensa en términos de atributos para describir estados, se pueden tener listas tabú para cada atributo y una función de aspiración que depende de cada atributo.

Si uno o más atributos pasan la prueba individual de aspiración, entonces se puede asumir que los demás atributos automáticamente también la pasan.

5.4 Intensificación y Diversificación

La idea de *intensificación* es buscar más es porciones del espacio que aparentan ser mejores o más prometedoras.

Cada determinado tiempo se puede realizar un proceso de intensificación.

Normalmente se re-inicia la búsqueda a partir de la mejor solución actual. Algunas opciones:

- Mantener fijos los componentes o atributos que parecen mejores.
- Cambiar el equema de vecindad.

Una extensión es considerar los comportamientos de los patrones producidos por una lista tabú.

Uno de los efectos de ésto es analizar por ejemplo movimientos oscilatorios evaluando lo atrayente de los movimientos dependiendo de la localización y dirección de la búsqueda.

Con ésto se puede por ejemplo especificar un número de movimientos necesarios en una cierta dirección antes de permitir algún retorno.

Las *funciones de memoria a mediano plazo* sirven para registrar y comparar atributos de las mejores soluciones obtenidas durante un período de búsqueda.

Los atributos comunes puedes servir para guiar nuevas soluciones a espacios en donde existan tales atributos.

Diversificación obliga buscar en áreas no exploradas.

Las *funciones de memoria a largo plazo* tratan de diversificar la búsqueda, empleando principios más o menos contrarios a los de memoria a mediano

plazo.

La idea es explorar regiones que contrastan fuertemente con las regiones exploradas hasta el momento.

No se trata de inyectar diversidad aleatoria, sino tomando en cuenta el proceso de búsqueda llevado hasta ese momento.

Para escapar de atractores fuertes se requiere de un esfuerzo adicional. Posibilidades:

1. Imponer restricciones más fuertes en las condiciones tabú para excluir un número más grande de movimientos.
2. Usar información de cuándo el proceso apunta hacia arriba y tratar de favorecer movimientos que apunten en esa dirección.
3. Penalizar movimientos que usan atributos muy usados en el pasado.

Una vez que se sale del atractor, se pueden "relajar" las condiciones.

Conceptualmente, lo que tratan de estimar es una distancia de escape del óptimo local.

Se pueden incorporar también elementos probabilísticos, para preferir movimientos con cierta probabilidad.

Algunos resultados muestran que el tamaño de la lista tabú entre 5 y 12 (y alrededor de 7) es adecuado.

Sin embargo, para algunos problemas se tiene que determinar cuál es la mejor.

5.5 The Reactive Tabu Search

La búsqueda tabú reactiva (RTS) adapta el tamaño de la lista a las propiedades del problema de optimización.

Se almacenan las configuraciones visitadas y sus números de iteración, por lo que se puede calcular el número de repeticiones de una configuración y el intervalo entre dos visitas.

El mecanismo de reacción básico aumenta la lista tabú cuando se repiten configuraciones y lo reduce cuando no se necesita aumentar.

Se añade un mecanismo de diversificación de memoria a largo plazo cuando se sospecha de un atractor fuerte.

Para entender los atractores, los mínimos locales se pueden ver como atractores en la dinámica de la política de máxima pendiente.

Por otro lado, también se pueden tener ciclos, que se repiten continuamente.

Una tercera posibilidad es que la trayectoria esté restringida a una cierta área del espacio de búsqueda (atractores caóticos).

Existe en sistemas dinámicos el concepto del *exponente de Lyapunov*. Si tenemos una función g que mapea un punto en el paso n a un punto en el paso $n+1$, $g^k(x)$ se define como el mapeo obtenido al interactuar g, k veces.

Si se empieza con configuraciones cercanas x_0 y $x_0 + \epsilon$, el exponente de Lyapunov λ se define por la relación:

$$\epsilon e^{n\lambda} \approx || \; g^n(x_0 + \epsilon) - g^n(x_o) \; ||$$

Si $\lambda > 0$, los puntos inciales divergen exponencialmente, y si las trayectorias se mantienen dentro de una región en el espacio, uno obtiene lo que se llama cáos determinístico.

La trayectoria parece aleatoria pero el sistema es determinístico. En tal caso, aunque no se tienen ciclos, sólo se visita una región pequeña del espacio.

El evitar ciclos no debe de ser la única meta, se tienen que continuar estimulando el descubrimiento de soluciones de mejor calidad.

Algoritmo:

1. Se evaluan todos los posibles movimiento elementales a partir de la configuración actual.

2. Se busca la última configuración y se decide si se hace un mecanismo de escape.

3. Si no hay escape, se realiza el mejor movimiento.

4. Si hay escape, el sistema entra en un ciclo de movimientos aleatorios cuya duración depende del promedio de los movimientos en ciclos detectados.

Cuando se hace una repetición, el mecanismo básico de reacción aumenta el tamaño de la lista tabú. Después de un número suficiente de repeticiones se elimina cualquier tipo de ciclo.

Esto no es suficiente para evitar trampas caóticas. Para ésto, se tiene otro mecanismo más lento que cuenta el número de configuraciones que son repetidas. Cuando el número es mayor a una cierta constante se entra al mecanismo de diversificación.

Por otro lado, también se tiene un proceso lento que reduce la lista tabú si el número de iteraciones es más grande que el promedio de movimientos entre ciclos desde el último cambio.

Cuando la lista tabú es tan grande que todos los movimientos son tabú y ningúno satisface el criterio de aspiración, entonces se reduce la lista tabú.

La estrategia de escape se basa en realizar un conjunto de movimientos aleatorios proporcionales al número promedio de movimientos entre ciclos.

Para evitar regresar a la región, todos los movimientos aleatorios se vuelven tabú.

En experimentos realizados, si se elimina el mecanismo de escape, el sistema frecuentemente queda atrapado y no encuentra la solución óptima.

5.6 Continuous RTS

La idea es proponer una estrategia híbrida:

- Optimizador local de alta precisión.
- Componente combinacional para descubrir regiones prometedoras.

Como el optimizador local es costoso solo se activa cuando existe un alta probabilidad de tener un buen óptimo local

5.6.1 Optimizador Local (*Affine Shaker*)

El algoritmo empieza con un punto inicial y una región de búsqueda que rodea al punto. Pasos:

1. Se genera un nuevo punto tentativo con probabilidad uniforme (aleatorio).
2. Se adapta la región de búsqueda de acuerdo al valor de la función de evaluación en el nuevo punto. Se comprime si el valor es mayor que el actual (peor) o se expande en caso contrario (mejor).
3. Si es un mejor punto, éste se vuelve el nuevo punto actual.

Para generar un nuevo punto se genera un valor aleatorio (δ). Si el nuevo punto ($x + \delta$) no es bueno, se explora su punto espejo ($x - \delta$).

El criterio de terminación se basa en un factor ϵ definido por el usuario (normalmente tomando comparaciones de pasos consecutivos).

5.6.2 Búsqueda por regiones

La región inicial de búsqueda está especificada por los rangos de un conjunto de variables independientes.

La región de búsqueda se divide en un árbol de cajas (*tree of boxes*).

Inicialmente se divide cada rango de cada variable en dos (produciendo 2^N cajas para N variables).

La unión de todas las hojas coincide con el epacio de búsqueda inicial.

Cada vez que se encuentran dos mínimos locales en una caja, el espacio se vuelve a subdividir.

Por lo que se pueden tener árboles a diferente profundidad en regiones diferentes.

En el proceso de búsqueda tabú, ahora en lugar de tener puntos, tenemos regiones, por lo que: (i) tenemos que evaluar el potencial de cada región, y (ii) el número de cajas crece con el tiempo (dinámico).

El proceso de búsqueda tabú tiene que identificar cajas promisorias rápidamente.

Una forma de evaluación simple es tomar un punto aleatorio en la región usando una distribución uniforme. Para evitar que se generen puntos coincidentales (muy malos o muy buenos), si se vuelve a visitar la región (caja) se regresa información acumulativa de la región.

Posibles esquemas:

1. Se regresa el promedio de los puntos visitados.
2. Se regresa el mínimo de los puntos visitados.

Debido a que el árbol puede tener hojas (cajas/regiones) de diferente tamaño no es directo identificar las regiones vecinas de un nuevo vecino generado.

Si el vecino generado no tiene una región ya generada, entonces se usa la región más pequeña generada que lo incluya.

Si se cubre más de una región, se selecciona una de ellas en forma aleatoria.

Los vecinos se consideran si no son regiones prohibidas (TL).

Si yá se exploró una región, ésta no se vuelve a explorar a menos que exista evidencia de un nuevo mínimo local.

En cuanto se identifica otro mínimo local, se divide la región en 2^N nuevas regiones.

Como puede verse la idea básica de búsqueda tabú es relativamente simple y lo que se ha hecho son extensiones para modificar dinámicamente la lista tabú o el esquema de vecindad para salir, como en los casos anteriores, de mínimos locales.

Por otro lado, las ideas de intesificación y diversificación también son temas de estudio para la mayoría de las técnicas que estamos viendo.