TEAM_GETPPID__
www.wuolah.com/student/TEAM_GETPPID__
32510

Metahuristica Tema 2.pdf

Resúmenes Metaheurística

3° Metaheurísticas

Grado en Ingeniería Informática

Escuela Politécnica Superior de Córdoba
UCO - Universidad de Córdoba

2019

Tema 2: Métodos basados en trayectorias

Contenido

1. Búsqueda basada en trayectorias

Este tipo de búsqueda se denomina así porque utilizan una sola solución durante el proceso de búsqueda, es decir, la solución describe una trayectoria desde la solución de partida hasta encontrar la solución final.

Algunos términos importantes:

- **Término local**: refleja el concepto de proximidad entre las soluciones alternativas del problema.
- **Soluciones vecinas**: todas las soluciones incluidas en el entorno de la solución actual, viene delimitado por un operador de generación de soluciones.

Este tipo de algoritmos realizan búsquedas locales en el espacio de búsqueda, en el entorno local de la solución actual para decidir como continuar la trayectoria.

Las búsquedas locales son procesos que dada una solución seleccionan de forma iterativa una solución de su entorno para continuar la búsqueda.

2. Métodos básicos de búsqueda local

Ingredientes de una búsqueda local

- Codificación y evaluación
- Selección de solución inicial
- Soluciones vecinas (vecindario o entorno)
- Estrategia de exploración del vecindario
    - Primera solución que mejore (escalada simple)
    - Mejor solución que mejore (escalada por la máxima pendiente)
- Criterio de parada

2.1 Escalada simple

La Idea de la búsqueda en escalada:

- Aplicar una simple mejora iterativa
- Guiado por la heurística y aleatoriedad de la función que genera sucesores
- No se permite recuperarse de un camino erróneo (no se mantiene un árbol de búsqueda)
- Puede verse como una escalada (ascenso o descenso)

```
M<- estado inicial
Iterar
        M'<-Generar solución vecina de M
        Si f(M') es mejor que f(M)
                M<-M'
Hasta que no haya más M'
Devolver M
```

En la formulación original de la escalada, las reglas de producción siguen un orden específico, con lo que el **orden de generación de vecinos es determinista**.

Existe una **versión estocástica** de la escalada, en la que se eligen las reglas de producción al azar.

2.2 Escalada por la máxima pendiente

La escalada por máxima pendiente se define como seleccionar mejor operación que suponga una mejora entre todas las vecinas a la solución actual.

```
M<- estado inicial
Iterar
        Generar todas las soluciones vecinas de M
        M'<-Mejor solución de entre todas las generadas
        Si f(M') es mejor que f(M)
                M<-M'
Hasta que no haya más M'
Devolver M
```

Estos dos métodos son muy probables de que se queden atrapados en óptimos globales, el método con máxima pendiente con una mayor probabilidad que la escalada simple. Se dirigen siempre a un estado mejor que el actual y no mantienen información de estados previos.

3. Metaheurísticas basadas en trayectorias

El problema con el que se encuentra la búsqueda local es lo explicado previamente y es que es muy probable que caigan en óptimos locales. Por ello se proponen soluciones para salir de estos óptimos locales:

- Permitir movimientos de empeoramiento de la solución actual
    - Ejemplo: Enfriamiento simulado, búsqueda Tabú, etc.
- Modificar la estructura de entornos
    - Ejemplo: Búsqueda Tabú, Búsqueda en entornos variables, etc.
- Volver a comenzar la búsqueda desde otra solución inicial
    - Ejemplo: Iterated local search, etc.

3.1 Enfriamiento simulado

Algoritmo de búsqueda local (búsqueda por entornos) con un criterio probabilístico de aceptación de soluciones basado en la Termodinámica. Evita que la búsqueda local finalice en óptimos locales permitiendo algunos movimientos hacia soluciones peores.

Consiste en generar aleatoriamente una solución cercana a la solución actual (o en el entorno de la solución) y la acepta como buena si consigue reducir una determinada función de costo, o con una determinada probabilidad de aceptación. Esta probabilidad de aceptación se irá reduciendo con el número de iteraciones y está relacionada también con el grado de empeoramiento del costo, es decir, en el algoritmo Simulated Annealing se pueden aceptar soluciones que empeoran la solución actual, solo que esta aceptación dependerá de una determinada probabilidad que depende de un parámetro, denominado temperatura.

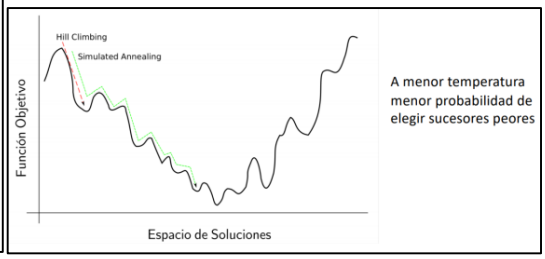
Inspirado en un proceso físico de enfriamiento controlado

- Se calienta un metal a alta temperatura y se enfría progresivamente
- Si el enfriamiento es correcto se obtiene la estructura de menor energía (mínimo global)

Comparación de términos:

- Estados del sistema – soluciones factibles
- Energía – coste
- Cambio de estado – solución en el entorno
- Temperatura – parámetro de control
- Estado congelado – solución heurística

```
Partimos de una temperatura inicial
mientras la temperatura no sea cero hacer
    // Paseo aleatorio por el espacio de soluciones
    para un numero prefijado de iteraciones hacer
        Enuevo ← Genera_sucesor_al_azar(Eactual)
        ΔE ← f(Eactual) − f(Enuevo)
        si ΔE > 0 entonces
            Eactual ← Enuevo
        sino
            con probabilidad e^(ΔE/T): Eactual ← Enuevo
        fin
    fin
    Disminuimos la temperatura
fin
```

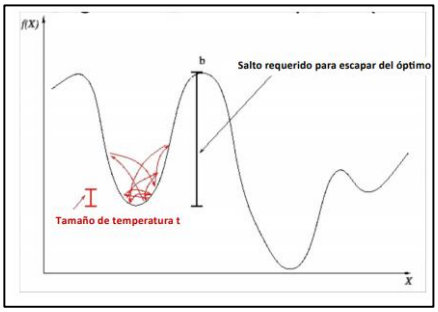


El proceso de enfriamiento es el mecanismo por el cual la temperatura va tendiendo a 0. Es decir, la probabilidad de aceptación de soluciones peores tiende a 0. El proceso de enfriamiento, por tanto, determina cómo se modifica la temperatura durante la ejecución del algoritmo. No es conveniente usar un valor fijo que sea

independiente del problema. Para converger al óptimo global se requiere un enfriamiento lento. Existen distintos tipos:

- Exponencial: $T = \alpha^{it} T_0$ / $T = \alpha T$
- Lineal: $\mathrm{T} = T_0 - \mathrm{n} \cdot \mathrm{it}$
- Boltzmann / Logarítmica : $\mathrm{T} = \frac{T_0}{(1+log(1+it))}$
- Cauchy : $\mathrm{T} = \frac{T_0}{(1+\mathrm{it})}$

Aunque este algoritmo ayude a no caer en óptimos locales, todavía es posible caer en ellos, por ello a vece se una técnica de recalentamiento que permite salir de óptimos locales.



3.2 Búsqueda Tabú

La búsqueda tabú aumenta el rendimiento del método de búsqueda local mediante el uso de estructuras de memoria: una vez que una potencial solución es determinada, se la marca como "tabú" de modo que el algoritmo no vuelva a visitar esa posible solución. Una lista tabú es una memoria de corto plazo que contiene las soluciones que fueron visitadas en el pasado reciente. Estas listas también se pueden aplicar por atributos en vez de por soluciones.

Las listas en memoria se usan para no repetir la trayectoria de búsqueda. Existen 2 tipos:

- Memoria a corto plazo: Evitan ciclos
- Memoria a largo plazo: Intensifican la búsqueda en buenas regiones y diversifican las búsquedas hacia nuevas regiones.

Procedimiento iterativo

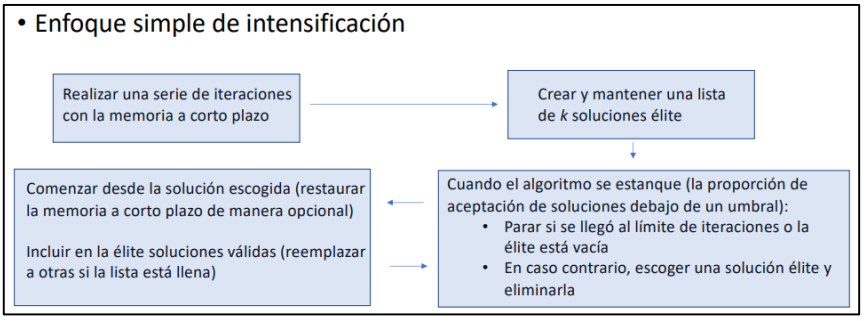
1. Solución actual
2. Definir vecindario
3. Evaluar vecindario
4. Elegir el mejor No-Tabú. El movimiento **siempre se acepta** a pesar de que la nueva solución sea peor que la anterior

La búsqueda tabú se vuelve mucho más potente incluyendo memoria de largo plazo y sus estrategias de reinicialización asociadas.

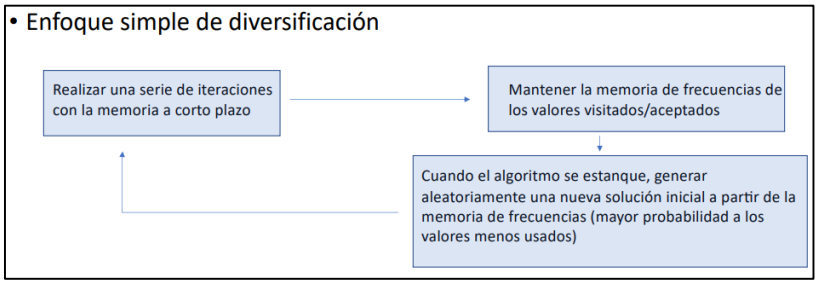
Podemos identificar estrategias de **intensificación** que se basan en la reinicialización de la búsqueda que efectúa un regreso a regiones atractivas del espacio de búsqueda. Se mantiene un registro de las mejores soluciones. También se puede introducir una medida de **diversificación** para asegurar que las soluciones registradas difieren unas de otras en cierto grado.

Existen cuatro variantes:

- Solución desde la que se reinicializa
    - Reanudar el proceso desde la mejor
    - Usar una pila de longitud limitada de las mejores y coger la cabeza de la pila
- Restauración de la memoria a corto plazo
    - Almacenar la memoria a corto plazo en el momento en el que se encontró: Restaurar
    - Borrar la memoria a corto plazo: Iniciar desde cero



Las estrategias de diversificación conducen hacia nuevas regiones del espacio de búsqueda no exploradas aún. La búsqueda se reinicializa cuando se estanca, partiendo de una solución no visitada. Solución a partir de la memoria de frecuencias, dando lugar a una mayor probabilidad de aparición de los valores menos habituales.



Es interesante un proceso de variación entre estrategias de intensificación y diversificación.

3.3 Búsqueda local iterativa

La base de este tipo de algoritmos es la aplicación repetida de un algoritmo de búsqueda local a una solución inicial que se obtiene por mutación de un óptimo local previamente encontrado. Se forman de 4 componentes: Solución inicial, procedimiento de búsqueda local, procedimiento de perturbación o modificación y el criterio de aceptación de a qué solución se aplica la perturbación.

```
Comienzo-ILS
        S_o ← Generar una solución inicial
        S ← Búsqueda local (S_o)
        Repetir
                S' ← Modificar (S, historia)
                S'' ← Búsqueda Local (S')
                S ← Criterio de aceptación (S, S'', historia)
                Actualizar MejorSolución
        Hasta (Condición de parada)
Devolver MejorSolución
Fin-ILS
```

3.4 Búsqueda por entornos variables (VNS)

Se basa en la idea de un cambio sistemático de entorno o vecindad dentro de una búsqueda local (aumentando el tamaño cuando la búsqueda no avanza).

Se basa en tres hechos:

- Un mínimo local en un entorno no lo es necesariamente en otro
- Un mínimo global lo es en todos los entornos
- En muchos problemas, los mínimos locales con los mismos o distintos entornos están relativamente cerca

```
Sea E_k (k = 1, ..., k_max) un conjunto finito de estructuras de vecindario
(entorno) preseleccionadas

Comienzo-VNS
    S <- Generar solución inicial
    k <- 1
    mientras (k <= k_max )
        S' <- Mutación en E_k (S)
        S'' <- Búsqueda Local (S')
        si S'' mejor que S entonces
            S <- S''; k <- 1
        si no
            k <- k+1
        finsi
    fin mientras
    Devolver S
Fin
```

3.5 GRASP (Greedy Randomized Adaptive Search Procedure)

Es un método multiarranque en el que cada iteración construye una solución greedy a la que aplicada una búsqueda local que toma dicha solución como punto de partida. Este proceso se repite varias veces y la mejor solución encontrada sobre todas las iteraciones GRASP se devuelve como la salida.

```
Procedimiento GRASP
  mientras (no se satisfaga el criterio de parada)
      S ← Construcción Solución Greedy ()
      S' ← Búsqueda Local (S)
      Actualizar (S', Mejor_Solución)
  fin mientras
  Devolver (Mejor_Solución)
FIN-GRASP
```

Este procedimiento iterativo se forma de: Algoritmo Greedy + Búsqueda local.