Fundamentos de Programación Entera

9. Heurísticas

Carlos Testuri – Fernando Islas

Departamento de Investigación Operativa – Instituto de Computación
Facultad de Ingeniería – Universidad de la República

2012–2025

Contenido

Motivación

Dada la complejidad de la mayoría de los problemas, una alternativa relevante de metodología de resolución son las heurísticas.
Una *heurística* es un método de resolución aproximado que busca obtener buenas soluciones utilizando una menor cantidad de recursos (tiempo o memoria).

Criterios de uso:

1. Se necesita una solución factible en poco tiempo.
2. La instancia es tan grande que, aunque se pueda formular mediante IP, la resolución mediante métodos exactos no es viable.
3. La instancia es tan compleja que la formulación mediante IP no es posible.
4. Para algunos problemas combinatorios complejos, son más eficientes que los métodos generales exactos.

Valoración de heurísticas

La simplicidad de las heurísticas tiene el compromiso de que no hay un mecanismo general de valorar las soluciones.

Criterios de valoración:

1. Existencia de criterio para medir la calidad de la solución en relación al óptimo.
2. Existencia de criterio para medir la calidad de la solución previo al uso de la heurística.
3. Existencia de criterio para medir en promedio la calidad de la solución de una clase de problemas previo al uso de la heurística.

Heurística ávida y búsqueda local

Una *heurística ávida* (greedy) construye una solución factible inicial eligiendo lo más ventajoso en cada paso.

Luego se trata de mejorar la solución inicial con una búsqueda en la vecindad de soluciones, *búsqueda local*.

Es un mecanismo general que debe adaptarse a las particularidades de cada problema; donde el proceso de adaptación puede tener multiples implementaciones.

Heurística ávida y búsqueda local: ejemplo 1/2

Dado el problema de mochilero binario

$$\begin{array}{rll} z = & \max & 17x_1 + 10x_2 + 12x_3 + 18x_4 \\ & \text{s.a} & 5x_1 + 3x_2 + 4x_3 + 6x_4 \leq 9 \\ & & x \in \mathbb{B}^4. \end{array}$$

Se tiene que las variables se encuentran ordenadas según valor de $\frac{c_j}{a_j}$ descendente.

1. Dado que $\frac{c_1}{a_1} = 3.4$ es el mayor y que hay capacidad de 9 unidades, se asigna $x_1 = 1$.
2. Luego, dado que $\frac{c_2}{a_2} = 3.3$ es el mayor y que quedan $9 - 5 = 4$, se asigna $x_2 = 1$.
3. Activar las restantes variables requiere mayor cantidad de la disponible $4 - 3 = 1$, por lo que se asignan $x_3 = x_4 = 0$.

Heurística ávida y búsqueda local: ejemplo 2/2

La solución ávida es $x^A = (1, 1, 0, 0)$ con valor objetivo $z^A = 27$.

Un criterio de búsqueda local puede ser poner en cero uno de los valores de x_1 o x_2 y continuar con el proceso usado en la solución inicial para las variables x_3 y x_4.

1. Si se establece $x_1 = 0$ y $x_2 = 1$ quedan $9 - 3 = 6$ unidades.
2. Dado que $\frac{c_3}{a_3} = 3$ y que quedan 6 unidades, se asigna $x_3 = 1$.
3. Activar la variable restante, x_4, requiere mayor cantidad que la disponible, $9 - 7 = 2$, por lo que se asigna $x_4 = 0$.

Obteniéndose la solución $x^B = (0, 1, 1, 0)$ con valor objetivo $z^B = 22$; la cual es inferior a z^A.
Si se establece $x_1 = 1$ y $x_2 = 0$ y se aplica el procedimiento de búsqueda se obtiene la solución $x^C = (1, 0, 1, 0)$ con valor objetivo $z^C = 29$; la cual es superior a z^A.

Heurística ávida: formalización

Dado el problema en términos de optimización combinatorial

$$\max_{S \subseteq N} \{c(S) : v(S) \leq r\}$$

En el problema del mochilero $c(S) = \sum_{j \in S} c_j$, $v(S) = \sum_{j \in S} a_j$ y $r = b$.

Heurística ávida

1. Sea $t := 1$, $S^1 := \varnothing$.
2. Elección de variable con mayor tasa marginal de valor por unidad de recurso: sea $j_t := \text{argmax} \frac{c(S^{t-1} \cup \{j_t\}) - c(S^{t-1})}{v(S^{t-1} \cup \{j_t\}) - v(S^{t-1})}$.
3. Si la solución S^{t-1} es factible, y el objetivo no ha aumentado, Parar con solución S^{t-1}.
4. En otro caso $S^t := S^{t-1} \cup \{j_t\}$. Si la solución es factible y el objetivo no aumenta o $t = t_{max}$, Parar con solución S^t.
5. En otro caso si $t = t_{max}$ y no hay solución factible Parar.
6. Establecer $t := t + 1$ e ir a paso 2.

Búsqueda local: formalización

En la búsqueda local se pone énfasis en la factibiblidad.
Se define una solución S, la *vecindad local* $Q(S)$ de la solución, y una *función objetivo ampliada* $f(S)$ que discrimina entre $c(S)$ cuando S es factible e infinita en otro caso.

Heurística de búsqueda local

1. Sea $t := 1$ y la solución inicial S^1.
2. Buscar $S^{t+1} \in Q(S^t)$ que cumple $f(S^{t+1}) > f(S^t)$; si no se encuentra Parar con solución S^t.
3. En otro caso establecer $S^{t+1} := S^t$.
4. Establecer $t := t + 1$ e ir a paso 2.

Búsqueda local avanzada

El criterio de búsqueda local básica solo permite sustituir la solución corriente por otras soluciones en su vecindad que mejoren el objetivo.
En el caso de existencia de múltiples óptimos locales, no permite indagar otros.

Una alternativa es permitir que a veces, en la búsqueda, las nuevas soluciones no sean tan buenas como la solución corriente; con la esperanza de poder indagar zonas de influencia de diferentes óptimos locales.

Este enfoque es utilizado por las heurísticas *búsqueda tabú* (tabu search) y *recocido simulado* (simulated annealing).

Búsqueda tabú

El riesgo de permitir que la nueva solución no sea tan buena como la corriente es que el proceso puede quedar atrapado en una circulación sobre una secuencia parcial de soluciones.
Para garantizar la convergencia se deben excluir soluciones (*tabú*). Para esto se mantiene una lista de las anteriores soluciones más recientes.

Búsqueda tabú

1. Inicialización de la lista
2. Obtención de solución inicial
3. Mientras que el criterio de parada no se alcance
   Seleccionar soluciones que no estén en la lista: $Q'(S) \subseteq Q(S)$.
   Sea $S' = \text{argmax}\{f(T) : T \in Q'(S)\}$
   Establecer $S := S'$ y actualizar la lista

Búsqueda tabú: parámetros

1. La selección de soluciones que no estén en la lista depende del tamaño de la vecindad $Q(S)$, una opción es elegir aleatoriamente un cantidad determinada por los recursos.
2. El tamaño de la lista es dependiente de los recursos disponibles y del problema.
3. El criterio de parada puede ser una cantidad determinada de iteraciones en las que no hay una mejora del objetivo de la mejor solución encontrada o del promedio de los objetivos de las soluciones de la lista.

Simulated annealing

La solución se elige en forma aleatoria. Una nueva solución sustituye a la corriente si tiene mejor valor objetivo; y con determinada probabilidad, también la sustituye aunque tenga peor valor objetivo.

La probabilidad de aceptar una solución con peor valor objetivo es inversamente proporcional a la diferencia de su valor objetivo con la solución corriente. Lo que permite escapar a la influencia de los óptimos locales.

Luego la probabilidad de aceptar peores soluciones decrece a medida que avanza el proceso, garantizando la convergencia del mismo.

Simulated annealing: algoritmo

Simulated annealing

1. Obtener solución inicial S.
2. Obtener temperatura inicial T y factor de reducción $r \in (0, 1)$.
3. Mientras $T \geq T_0$ (cond. de parada)
   Seleccionar aleatoriamente un vecino S' de S.
   Sea $\Delta = f(S') - f(S)$.
   Si $\Delta \geq 0$, entonces $S := S'$.
   sino $S := S'$ con probabilidad $e^{\Delta/T}$.
   Establecer $T := rT$

Búsqueda local avanzada: propiedades

1. *Continuidad del vecindario*
   La estructura debe ser tal que sea fácil pasar de la solución corriente a la nueva.

2. *Diversificación*
   El grado de viabilidad de moverse entre distintas áreas de influencia de óptimos locales. Ejemplo: una larga lista tabú o alta temperatura inicial de annealing.

3. *Intensificación*
   Característica de incrementar la búsqueda en áreas promisorias.

Algoritmos genéticos

Emulan la evolución en seres vivos. En vez de tratar de mejorar una solución por vez, lo hace con un conjunto de soluciones (población) que evoluciona (cruzamiento y mutación) con cada iteración (generación).

Algoritmos genéticos

Mientras no se determine mejor solución

1. Evaluar el valor (fitness) de los individuos.
2. Seleccionar los ancestros a aparear en base a su valor.
3. Cruzar pares de ancestros seleccionados produciendo dos descendientes.
4. Mutar algunos de los descendientes modificándolos aleatoriamente.
5. Selección de individuos, en base a su valor, que reemplazan parcialmente o toda la población corriente.

Algoritmos genéticos: instrumentación

1. La evaluación de los individuos tiene en cuenta el objetivo y las restricciones del problema con funciones de infactibilidad.
2. La selección de los individuos a cruzar puede hacerse en función del valor de cada individuo normalizado con respecto al resto de la población.
3. El cruzamiento depende de la estructura de representación del individuo. En general se representa con una secuencia de un alfabeto dado (binario, etc) que en el cruzamiento puede tener uno o multiples puntos de intercambio.
4. La mutación es la modificación aleatoria puntual de la estructura de un individuo.