

Inteligencia Artificial

Ayudantía para certamen 2

Fernanda Weiss
fernanda.weiss.13@sansano.usm.cl

2018-2

1 Técnicas de búsqueda incompleta

1.1 Algoritmos de búsqueda single-point

Los algoritmos de búsqueda incompleta son algoritmos que intenta encontrar una buena solución, pero no garantiza encontrarla (o encontrar la mejor) ni tampoco son capaces de determinar si el problema no tiene solución. Recorren el espacio de búsqueda con alguna heurística, de forma miope o estocástica, con la esperanza de encontrar el mejor óptimo local (óptimo global). Esto les permite alcanzar una eficiencia que no es posible de alcanzar por los algoritmos de búsqueda completa, presentando un mejor rendimiento en los problemas más complejos que las técnicas de búsqueda completa que realizan una costosa búsqueda exhaustiva.

Definiciones generales:

- Instancia: un par (S', f) . S' representa el espacio de soluciones y f una función de evaluación que señala la calidad de la solución.
- Vecindario de la solución: $N(s)$. Conjunto de soluciones alcanzables desde s por medio de una pequeña modificación o movimiento.
- Solución candidata: posible instanciación para el problema, pudiendo ser factible o infactible.
- Representación: forma de estructurar la solución. Típicamente es un vector donde cada elemento representa una variable.
- Movimiento: tipo de cambio utilizado para reparar una solución. El vecindario depende del movimiento y NO al revés. La inserción y permutación son movimientos válidos que dependen de la representación y el tratamiento que se está dando a las restricciones.
- Explotación o Intensificación: enfatiza el proceso de búsqueda en el vecindario de la solución actual, para encontrar el óptimo local en el área.
- Exploración o Diversificación: estrategias de salto en áreas inexploradas del espacio de búsqueda. Se aplican cuando (o para evitar que) la búsqueda se quede estancada en un óptimo local.
- Óptimo local: aquella solución que tenga mejor calidad que todas las soluciones del vecindario.
- Óptimo global: aquella solución factible que tenga mejor calidad que todas las soluciones del espacio de búsqueda.

1.1.1 Algoritmos de construcción

- Construyen una solución de forma incremental.
- Comienza con una solución inicial vacía o con un solo componente. Es el punto de partida desde dónde comienza a construir la solución.
- Tiene una función miope, la cual toma decisiones localmente óptimas según la función de evaluación.
- Algoritmo Greedy o Goloso: añade a la solución el componente que produce el mayor beneficio local (función miope).
- Son rápidos y sencillos, pero no siempre encuentran la mejor solución y por lo general son deterministas por lo que el número de soluciones que encuentran es limitado y es muy probable que no sean capaces de encontrar el óptimo.

1.1.2 Algoritmos de reparación

- Encuentran solución por un método iterativo que va mejorando la solución.
- Comienzan con una solución que es generada aleatoriamente o por medio de un algoritmo de construcción y luego la intentan mejorar.

Hill-Climbing

Algorithm 1 Hill Climbing con Mejor Mejora

```

1:  $local \leftarrow FALSE$ 
2:  $s_c \leftarrow$  seleccionar una solución al azar
3: repeat
4:   asignar a  $s_n$  la solución de mejor calidad en  $N(s_c)$ 
5:   if  $f(s_n)$  es mejor que  $f(s_c)$  then
6:      $s_c \leftarrow s_n$ 
7:   else
8:      $local \leftarrow TRUE$ 
9:   end if
10: until local
  
```

Busca ir mejorando el valor de la función de evaluación de una solución candidata (generada aleatoriamente o arbitrariamente), realizando reparaciones. Para ello construyen el vecindario de la solución actual y seleccionan una solución que sea de mejor calidad. Cabe notar que no siempre será posible construir el vecindario completo en cada paso, por lo que se deberán definir criterios para cada vecindario.

- Queda fácilmente atrapado en óptimos locales.
- Es estocástico, puede llegar a un resultado distinto dependiendo de la solución inicial. Sin embargo, dos soluciones iniciales distintas pueden llegar al mismo resultado.
- No asegura un óptimo global
- Selección según alguna mejora: el primer vecino que mejora la calidad de la solución es aceptado inmediatamente.
- Selección según mejor mejora: se construye un vecindario según el movimiento, y luego, de ese vecindario, se elige al vecino de mejor calidad.

Hill-Climbing con Restart

Algorithm 2 Hill Climbing con Restart

```

1:  $t \leftarrow 0$ 
2: inicializar  $s_{best}$ 
3: repeat
4:    $local \leftarrow FALSE$ 
5:    $s_c \leftarrow$  seleccionar un punto al azar
6:   repeat
7:     asignar a  $s_n$  la solución de mejor calidad en  $N(s_c)$ 
8:     if  $f(s_n)$  es mejor que  $f(s_c)$  then
9:        $s_c \leftarrow s_n$ 
10:    else
11:       $local \leftarrow TRUE$ 
12:    end if
13:  until local
14:   $t \leftarrow t + 1$ 
15:  if  $s_c$  es mejor que  $s_{best}$  then
16:     $s_{best} \leftarrow s_c$ 
17:  end if
18: until  $t = MAX\_ITERATIONS$ 
  
```

Le agrega la capacidad de reinicio a Hill-Climbing. Cuando el algoritmo se queda atrapado en un óptimo local se reinicia desde otro punto. Esto permite seguir explorando el espacio de búsqueda. Sin embargo, es posible que se quede estancado en el mismo óptimo local si es que comienza en un vecindario que ya hubiese visitado en el pasado, perdiendo tiempo de búsqueda e información valiosa con el reinicio pues se pierde la solución que había estado construyendo. Por otro lado, es posible que el algoritmo nunca llegue al óptimo global debido a como selecciona la siguiente solución.

Tabu search

Búsqueda que permite la aceptación de soluciones de peor calidad permitiendo así evitar o escapar de óptimos locales. Restringe la búsqueda al clasificar ciertos movimientos/soluciones como prohibidos, para evitar caer en soluciones recientes y prevenir el generar ciclos durante el proceso de búsqueda. Para ello agrega estos movimientos/soluciones en una lista tabú. Esta lista generalmente se llena en forma FIFO.

- Lista tabú que guarda movimientos para impedir ciclos.
- La aceptación de soluciones de peor calidad es un mecanismo de exploración.
- El tamaño de la lista tabú permite ajustar el nivel de diversificación. El tamaño puede variar a lo largo del proceso de búsqueda.

Algorithm 3 Tabu Search

```

1: Inicializar  $s_c$  al azar
2: Inicializar  $tabu_{list}$  como lista vacia
3: Inicializar  $s_{best}$ 
4: repeat
5:    $s_v \leftarrow$  desde el  $N(s_c)$ , siendo este punto el que satisface el criterio de aspiración o el mejor punto no tabú
6:    $s_c \leftarrow s_v$ 
7:   Actualizar  $tabu_{list}$ 
8:   if  $s_c$  es mejor que  $s_{best}$  then
9:      $s_{best} \leftarrow s_c$ 
10:  end if
11: until criterio de termino

```

- Si la lista es larga habrá una gran diversificación pero podría llegar a restringir el tamaño de los vecindarios, y por lo tanto, la búsqueda y las posibilidades de encontrar el óptimo.
- Si la lista es corta aumenta la posibilidad de que se formen ciclos y que se visite varias veces el mismo espacio.
- Se puede introducir un Criterio de Aspiración que permita aceptar soluciones/movimientos tabú.
- Un ejemplo sencillo es si es que una solución candidata, a pesar de ser tabú, es mejor que la mejor encontrada, entonces se acepta.

Simulated Annealing o Recocido Simulado

Simula el enfriamiento de ciertos materiales, donde al estar a altas temperaturas sus partículas se agitan y a medida que desciende la temperatura van acomodándose. El algoritmo tiene una variable de temperatura que define la probabilidad que se acepte una peor solución favoreciendo la exploración. A medida que el algoritmo avanza esta temperatura va decreciendo evitando aceptar peores soluciones, y por ende, favoreciendo la explotación.

Pasos a seguir:

1. Inicializar T en un valor alto.
2. Inicializar solución inicial, aleatoria.
3. Generar solución cualquiera del vecindario, de forma aleatoria.
4. Si el candidato tiene mejor calidad se acepta inmediatamente. Si no, se hace una prueba con probabilidades:
 - Número aleatorio $< P$ entonces se acepta.

- $P = e^{\frac{\Delta Evaluation}{T}}$
- $\Delta Evaluation = \text{calidad solución nueva} - \text{calidad solución actual}$. Para un problema de maximización.
- $\Delta Evaluation = \text{calidad solución actual} - \text{calidad solución nueva}$. Para un problema de minimización.

5. Repetir los pasos 3 y 4 una cierta cantidad de veces.

6. Actualizar T.

7. Si no se ha cumplido el criterio de término, volver al paso 3.

Se utilizan enfriamientos sucesivos cada cierto número de iteraciones y recalentamientos periódicos para escapar de óptimos locales.

1.2 Algoritmos Genético

El algoritmo genético es un algoritmo que imita el proceso evolutivo de la selección natural de los seres vivos. El algoritmo genera una población de soluciones que pasan a través de 4 fases principales: selección, cruzamiento, mutación y elitismo.

Representación

Para poder hacer uso de un algoritmo genético es necesario una representación para las soluciones para llevar a cabo las fases de cruzamiento y mutación. Esta representación puede ser bien distinta al modelo. Algunas representaciones nos permitirán evitar soluciones infactibles.

La representación usada por el algoritmo genético estándar es la Representación Binaria donde las soluciones son representadas por bits. La cantidad de bits define el nivel de precisión disponible de obtener con la representación. Más precisión implica una mayor cantidad de bits.

Dos conceptos asociados a las soluciones y la representación son el de genotipo y fenotipo:

- Genotipo: es lo que se transmite. Caracteres hereditarios definidos por un conjunto de genes. No cambian. Por ejemplo, la representación.
- Fenotipo: no se transmite. Apariencia física, características observables y medibles. Pueden cambiar en el tiempo. Por ejemplo, la calidad en la función de aptitud.

Inicialización

Se crea una población inicial de soluciones. Esta población inicial es aleatoria y en lo posible debe estar sembrada con individuos diversos para favorecer la exploración.

Evaluación

Durante la etapa de evaluación se hace uso de la función de aptitud para determinar la calidad o el fitness de cada solución para su posterior selección. La etapa de evaluación debe hacerse después de la inicialización y luego en cada iteración entre las fases de mutación y elitismo.

Selección

Se seleccionan los individuos más aptos para ser transformados, pero no necesariamente solo los mejores. Esto se regula con la Presión de selección. Cuando la presión es alta, se tienden a elegir los mejores candidatos, favoreciendo la explotación. Si la presión es baja, es más probable que se admitan candidatos no tan buenos favoreciendo la exploración.

Tipos de selección:

- Ruleta: selección proporcional a la función de aptitud. Ponderación respecto a su calidad o fitness. La ruleta tiene alta presión de selección cuando hay gran variedad en calidad. Probabilidad de que un individuo sea seleccionado (en un problema de maximización): $P_i = \frac{f_i}{\sum_{j=1}^N f_j}$ Con f_i el desempeño del i-esimo individuo, y n el tamaño de la población.

Ejemplo:

$n = 6$

Individuo	Desempeño	Probabilidad
1	2	$\frac{2}{24} = 8,33\%$
2	12	$\frac{12}{24} = 50,00\%$
3	2	8,33%
4	1	4,17%
5	6	25,00%
6	1	4,17%

El problema con este método, es que si una fracción de la población posee una media de desempeño excesivamente alta, tendrán mucha más probabilidades de ser elegidos y por ende se perjudica la exploración. En el ejemplo, es posible que la mitad de las soluciones seleccionadas sean el individuo 2, perdiendo la información que podrían haber otorgado las soluciones con peor desempeño.

- Ranking: Los individuos se ordenan según su medida de desempeño en un ranking. En base a su posición en el ranking se define una ponderación con la que se obtiene la probabilidad de que sea seleccionado.

Ejemplo:

$n = 6$

Individuo	Desempeño	Rank	Probabilidad
I_2	12	6	$\frac{6}{23} = 26,1\%$
I_5	6	5	21,7%
I_1	2	4	17,4%
I_3	2	4	17,4%
I_4	1	2	8,7%
I_6	1	2	8,7%

Para obtener la probabilidad de selección se divide el lugar del ranking por la sumatoria de los puestos. Por ejemplo, el individuo más alto en el ranking anterior, está en el ranking 6. Se divide por $6 + 5 + 4 + 3 + 2 + 1 = 21$, que es el total de puestos, con una probabilidad final de ser elegido de : $\frac{6}{21} = 28\%$. En este caso, el desempeño no influye tanto por que sirve solo para armar el ranking, y por ende es menos discriminador con las soluciones de peor calidad en poblaciones muy diversas.

- K-Torneo: Se selecciona un conjunto de soluciones al azar y entre ellas se elige la que tenga mejor calidad. K define el tamaño del torneo, es decir, la cantidad de soluciones que se tomaran cada vez para comparar. Mientras K sea más alto, más alta será la presión de selección pues aumenta la posibilidad de seleccionar las mejores soluciones. Puede ayudar al rendimiento del algoritmo evitando evaluar soluciones que no sean elegidas por el operador para su comparación.

Cruzamiento

Durante esta etapa las soluciones se cruzan, es decir, se usan mezclas de información de las soluciones para generar una nueva población. Para mezclar las soluciones se hace uso de un operador de cruzamiento, normalmente binario (dos soluciones). Esta es una etapa de explotación.

Existe una probabilidad de cruzamiento que es un parámetro del algoritmo. Los hijos heredan algunas características de cada padre.

Cabe notar que ciertos operadores de cruzamiento pueden producir soluciones infactibles con algunas representaciones, por lo que un cambio de representación o una codificación antes de aplicar la operación durante esta etapa permitiría evitar esta situación. Por otro lado, ciertos operadores de cruzamiento pueden evitar la generación de soluciones infactibles.

Cruzamiento en un punto: se selecciona un punto al azar y se corta en ese punto generándose dos partes. Se usa la primera mitad de uno de los padres y la segunda mitad del otro para generar un hijo. Este cruzamiento es usado en el algoritmo genético estándar. Se generan dos hijos.

Mutación

La mutación es una operación que permite entregarle variabilidad al algoritmo. Es una probabilidad de un pequeño cambio aleatorio en un individuo. Generalmente el operador de mutación que se utiliza es Bit-flip, que cambia el valor del bit. Existe una probabilidad de mutación como parámetro del algoritmo. Esta probabilidad será para cada gen.

- La probabilidad P determina con cuanta frecuencia mutan los genes.

- Debe permitir alcanzar cualquier parte del espacio de búsqueda.
- Debe idealmente producir soluciones factibles.

Elitismo

Se busca en la población el mejor individuo y este se guarda de forma de almacenar la mejor solución encontrada hasta el momento. Como esta solución contiene información valiosa debido a su alta calidad, durante esta fase se vuelve a introducir en la población reemplazando a algún individuo.

Esta fase de elitismo puede también complementarse haciendo uso de la peor solución encontrada de forma de favorecer la exploración.

Diseño

Para evitar convergencia prematura en un algoritmo genético, se puede:

- No tener una presión de selección muy elevada.
- No tener una probabilidad de mutación muy baja.
- Variar la probabilidad de mutación durante la ejecución: alta al principio y baja al final.
- No tener un tamaño de población ineficiente o muy pequeña para el problema

1.3 Ejercicio

La Universidad tiene que panificar un evento que consiste en n conferencias. Para cada conferencia se conoce la hora de comienzo y la de finalización fijada por los exponentes. Se le ha pedido que planifique las n conferencias distribuyéndolas entre las distintas salas disponibles, de forma que no haya dos conferencias en una misma sala al mismo tiempo. El objetivo es minimizar el número de salas utilizadas, para así causar el menor trastorno al resto de las actividades académicas.

Proponga una representación, función de evaluación y una función miope que le permita utilizar un algoritmo constructivo para encontrar soluciones para el problema. Parta desde una solución vacía.

Respuesta

- Representación: $X_c = \text{Sala asignada a la conferencia } c$
 $c \in [0, n]$
- Función de evaluación: Minimizar la cantidad de salas usadas más penalización por topes.
- Función miope: Ordenar las conferencias por hora de inicio. Luego tratar de asignar las conferencias a las salas ya utilizadas como primera opción.

1.4 Ejercicio

El problema 2D Strip Packing consiste en acomodar un conjunto de objetos rectangulares en una tira de ancho finito y alto infinito. La idea es acomodarlos de tal forma que ocupen el mínimo espacio posible. Los objetos van siendo colocados de manera ortogonal y pueden ser rotados, y los lados de los rectángulos son menores que el ancho de la tira. Señale:

- Una representación.
- Operador de cruzamiento.
- Operador de mutación.

Tip: Considere que cuenta con una heurística que permite definir la posición donde se coloca un rectángulo en base a rectángulos previamente colocados.

Respuesta

Representación: Se pueden usar dos vectores. Con la heurística del tip podemos simplemente señalar el orden de los rectángulos. Primero consideramos R el conjunto de los rectángulos con R_i siendo el i -ésimo rectángulo del conjunto. Así, tenemos un vector indicando el orden de inserción de los rectángulos y otro indicando si un rectángulo se encuentra rotado o no:

X con $X_j \in R$ siendo el rectángulo R_i que entrara en la posición j
 Y con $Y_j \in 0, 1$ representando si el rectángulo R_i se encuentra rotado

Cruzamiento: Como la representación indica el orden en que se ingresan los rectángulos usar el operador de cruzamiento en un punto de forma directa podría generar inconsistencias. Sin embargo, podemos codificar la representación a otra antes de llevar a cabo un cruzamiento, pudiendo hacer uso de la Lista de Referencia. El vector que indica la orientación de los rectángulos no esta sujeto a este problema por lo que podemos usar el mismo operador.

Otro operador de cruzamiento sería el siguiente: se toman las dos soluciones y se selecciona un punto de corte. Para generar un hijo se toma la primera mitad de uno de los padres y luego se van agregando los rectángulos que faltan en el mismo orden que se encuentran en el segundo padre.

Mutación: Para el operador de mutación podemos aplicar transformaciones distintas para cada vector. Para el vector de orden de inserción se puede hacer un swap intercambiando la posición de inserción de dos rectángulos. Para el vector de rotación se puede usar el operador bit-flip.

1.5 Ejercicio

El Traveling Tournament Problem trata la planificación de un torneo de doble eliminación round-robin (DRR) jugado por n equipos donde cada equipo juega contra otro dos veces (de visita y de local). Los datos de entrada contemplan el número de equipos y una matriz D simétrica de $n \times n$, donde D_{ij} indica la distancia o el coste de viaje entre las ciudades locales de los equipos i y j . El problema consiste en encontrar una planificación de las rondas del torneo de tal forma de minimizar el tiempo de viaje total de los equipos cumpliendo las siguientes restricciones:

- No más de tres partidos consecutivos de local o de visita
- Un partido de los equipos T_i y T_j con T_i de local no puede ser seguido por el partido entre los mismos equipos con T_j de local.

Indique una representación y al menos tres operadores de mutación para esta.

Respuesta

Representación: Para la representación hacemos uso de una simplificación. Consideramos el problema como el Mirrored Tournament Problem, donde el torneo se divide en dos partes. Las primeras $n - 1$ rondas cada equipo juega contra todos los otros, y en la segunda parte vuelve a jugar contra ellos en el mismo orden solo que con los partidos de local y visita intercambiados. Luego declaramos una matriz M de $n \times n - 1$, donde la casilla M_{ij} indica quien será el oponente del equipo T_i en la ronda j . Si el número es positivo, quiere decir que T_i juega de local, sino que juega de visita.

Operadores de mutación: Entre los tres posibles operadores de mutación tendremos:

- Swap Local-Visita: Intercambia los roles de local y visita para algún partido entre dos equipos
- Swap Equipos: Intercambia la planificación de partidos de dos equipos. Es decir, los oponentes del equipo T_i pasaran a ser los de T_j y viceversa. Por razones obvias se evita cambiar el partido en donde se enfrentan los dos equipos.
- Swap Juegos: este operador consiste en elegir un juego arbitrario y forzar a jugarlo en una ronda, llevando a cabo todos los cambios necesarios para evitar que se juegue un mismo juego más de una vez en una misma ronda.

1.6 Ejercicio

Responda Verdadero o Falso según la aseveración y justifique:

1. El mecanismo que usa tabú search para escapar de óptimos locales es la lista tabú.
 F. El mecanismo para escapar de óptimos locales es aceptar soluciones de peor calidad. La lista tabú sirve para evitar ciclos. Las listas tabu largas promueven la exploración de áreas más extensas del espacio de búsqueda.
2. Siempre que se aplica restart en un algoritmo Hill-Climbing se está ayudando a explorar más.
 F. En general si, pero puede que un restart nos deje en el mismo lugar, por ende no siempre nos ayuda a explorar.
3. Es posible utilizar un método de sintonización para cambiar el largo de la lista durante la búsqueda de tabú search.
 F. Los métodos de sintonización son para setear parámetros fijos previos a la ejecución. Los métodos de control son los que permiten modificar elementos sobre la marcha.

4. Hill-climbing con alguna mejora es más eficiente que hill-climbing con mejor mejora, pero no asegura encontrar el óptimo global.
F. Ninguno asegura encontrar el óptimo global. La generación de vecindarios por “alguna mejora” busca reducir los costos de generación de vecindarios de gran tamaño.
5. En ningún caso un algoritmo genético podrá determinar que la solución encontrada será el óptimo global del problema.
V. En general el algoritmo encuentra óptimos locales, pero si la calidad de la solución óptima es conocida a priori el algoritmo podría saber que ha llegado al óptimo. Esto puede darse en el caso de los Problemas de Satisfacción de Restricciones puesto que la función de aptitud busca minimizar la cantidad de restricciones no satisfechas y se sabe que el óptimo se dará cuando sea cero.
6. Los operadores de cruzamiento exploran.
F. Los operadores de cruzamiento normalmente explotan pues utilizan información de soluciones ya encontradas.
7. El operador k-torneo ofrece a todas las soluciones de la población una oportunidad de resultar seleccionadas.
F. Si existe una solución que tiene una peor calidad que todas las demás soluciones, nunca será seleccionada por el operador k-torneo.
8. El cruzamiento en un punto no es útil para la resolución de problemas de optimización con restricciones.
V. En general, pero en algunos casos el cruzamiento en un punto se puede usar en conjunto con ciertas representaciones o codificaciones para evitar problemas con las restricciones.
9. La mutación de un algoritmo genético estándar es un algoritmo de búsqueda local.
F. La mutación es un operador, un pequeño cambio a una solución, no es una técnica de búsqueda en sí.