Metaheurísticas

Seminario 4. Técnicas basadas en trayectorias

1. Trayectorias Múltiples
   - Esquema General del Algoritmo BMB y ILS
   - Un Algoritmo ILS para los problemas de prácticas
   - Algoritmo GRASP
2. Trayectorias Simples
   - Esquema General del Algoritmo de Enfriamiento Simulado
   - Un Algoritmo de Enfriamiento Simulado para los problemas de prácticas

Procedimiento BMB

Comienzo-BMB

Repetir

S ← Generar-Solución-Aleatoria

S' ← Búsqueda Local (S)

Actualizar (*Mejor_Solución*, S')

Hasta (Condiciones de terminación)

Devolver *Mejor_Solución*

Fin-BMB

Procedimiento ILS

Comienzo-ILS

S_0 ← Generar-Solución-Inicial

S ← Búsqueda Local (S_0)

***Mejor_Solución* ← S**

Repetir mientras !(Condiciones de terminación)

S' ← Modificar (S, historia) **// Mutación**

S'' ← Búsqueda Local (S')

Actualizar (*Mejor_Solución*, S'')

S ← Criterio-Aceptación (S, S'', historia)

Devolver *Mejor_Solución*

Fin-ILS

Procedimiento ILS

Comienzo-ILS

S_0 ← Generar-Solución-Inicial

S ← Búsqueda Local (S_0)

***Mejor_Solución* ← S**

Repetir mientras !(Condiciones de terminación)

S' ← Modificar (Mejor_Solución) **// Mutación**

S'' ← Búsqueda Local (S')

Actualizar (*Mejor_Solución*, S'')

Devolver *Mejor_Solución*

Fin-ILS

ILS para ambos problemas

- Representación: vector de enteros.
- Solución inicial: aleatoria
- Operador de mutación: Cada vez que se muta, escogemos un 20% de los elementos, para provocar un cambio brusco, mínimo 2 elementos.
- Algoritmo de búsqueda local: dos variantes: la BL modificada de la Práctica 1 y el ES de esta misma práctica
- Criterio de aceptación: se sigue el “criterio del mejor”, siempre se aplica la mutación sobre la mejor solución encontrada hasta ahora

Procedimiento GRASP

Procedimiento GRASP

Repetir Mientras (no se satisfaga el criterio de parada)

S ← Construcción Solución Greedy Aleatorizada ()

S' ← Búsqueda Local (S)

Actualizar (S', *Mejor_Solución*)

Devolver (*Mejor_Solución*)

FIN-GRASP

Procedimiento GRASP

Construcción Solución Greedy Aleatorizada

- En cada iteración de su proceso constructivo de la solución, un algoritmo greedy básico:
  - construye una lista con los candidatos factibles (las posibles componentes a escoger de acuerdo con la solución construida hasta el momento y las restricciones del problema): **Lista de Candidatos (LC)**,
  - los evalúa de acuerdo a una función de selección (que mide su calidad/preferencia para ser escogidos), y
  - selecciona siempre el candidato de **mejor calidad** de la LC
- Los algoritmos GRASP añaden **aleatoriedad** al procedimiento anterior. La única diferencia es que en cada iteración:
  - No se consideran todos los candidatos posibles sino sólo los de mejor calidad: **Lista Restringida de Candidatos (LRC)**. El tamaño de esa lista puede ser fijo o variable en función de un umbral de calidad
  - El elemento seleccionado se escoge **aleatoriamente** de la RCL para inducir diversidad, independientemente de la calidad de los candidatos

Procedimiento GRASP para el MDD

- Nuestro algoritmo GRASP estará basado en el greedy usado como algoritmo de comparación hasta ahora.
- Se recuerda el esquema Greedy:

1. $Sel = \varnothing$
2. Calcular la dispersión a partir de las distancias de cada elemento al resto: , incluir el elemento s_{i*} que la maximice en Sel: $Sel = Sel \cup \{s_{i*}\}$ y eliminarlo de S: $S = S - \{s_{i*}\}$

while ($|Sel| < m$)

3. Calcular la dispersión no seleccionados, $s_i \in S$, del conjunto de elementos seleccionados Sel:
4. Incluir el elemento s_{i*} que la minimice en Sel: $Sel = Sel \cup \{s_{i*}\}$
5. Eliminar el elemento s_{i*} seleccionado de S: $S = S - \{s_{i*}\}$

end while

Procedimiento GRASP para el MDD

- La *LC* incluye todos los elementos no seleccionados: *S*-*Sel*. Los candidatos con menor dispersión al resto de elementos seleccionados hasta el momento (*Sel*) son preferibles

- La *LRC* es de tamaño variable e incluye todos los elementos no seleccionados cuya dispersión calculando la distancia a los actualmente seleccionados es mayor o igual que el umbral de calidad $\mu = d_{min} + \alpha \cdot (d_{max} - d_{min})$, donde d_{min} es la menor dispersión de los candidatos de *LC* y d_{max} la mayor dispersión.

- Se escoge aleatoriamente un elemento candidato de la *LRC* y se añade a la solución parcial $Sel \leftarrow Sel \cup \{s_{i*}\}$

- En cada nuevo paso del algoritmo hay que actualizar la *LC*, eliminando los elementos seleccionados en el paso anterior, y construir la nueva *LRC* recalculando las dispersiones para los candidatos factibles restantes y aplicando el umbral para filtrar los candidatos a emplear

- El proceso constructivo termina tras *m*-1 pasos

Procedimiento GRASP para el MDD

1. $Sel = \oslash$
2. Calcular aleatoriamente dos nodos, y guardar el resto de nodos posibles en S.

while ($|Sel| < m$)

3. Calcular la dispersión a partir de las distancias de cada elemento al resto, $s_i \in S$, al conjunto de elementos seleccionados *Sel*, obteniendo LR
4. Elegir aquellos con mejor dispersión según la fórmula y obtener *LRC*;
5. Escoger aleatoriamente un elemento s_{i*} de *LRC.*
6. Incluir s_{i*} en *Sel*: $Sel = Sel \cup \{s_{i*}\}$
7. Eliminar s_{i*} de *S*: $S = S - \{s_{i*}\}$

end while

Procedimiento GRASP para el SNIMP

- La *LC* incluye todos los elementos no seleccionados: *S-Sel*. Los candidatos con mayor distancia acumulada a los elementos seleccionados hasta el momento (*Sel*) son preferibles

- La *LRC* es de tamaño variable e incluye todos los elementos no seleccionados cuya dispersión calculando la distancia a los actualmente seleccionados es mayor o igual que el umbral de calidad $\mu = h_{max} - rand() \cdot (h_{max} - h_{min})$, donde h_{min} es la menor heurística de los candidatos de *LC* y h_{max} la mayor dispersión, y rand() es un número aleatorio entre 0 y 1 (distinto en cada paso).

- Se escoge aleatoriamente un elemento candidato de la *LRC* y se añade a la solución parcial $Sel \leftarrow Sel \cup \{s_{i*}\}$

- En cada nuevo paso del algoritmo hay que actualizar la *LC*, eliminando los elementos seleccionados en el paso anterior, y actualizando la heurística, Eliminando para cada nodo que lo tuviese de vecino los vecinos del nodo ya insertado en *Sel*.

- El proceso constructivo termina tras *m*-1 pasos

Procedimiento GRASP para el SNIMP

1. *Sel*=⊘
2. Calcular las heurística de todos los nodos, H.
3. Escoger el primer nodo aleatoriamente en *Sel*, y eliminarlo de *S: S* = *S* – s_{i*}.

while (|*Sel*|<*m*)

3. Actualizar H eliminando para cada nodo n vecino s_{i*} sus vecinos:

$$h_j = h_j - |N_u^+| \text{ para todo j es vecino de i}$$

4. Actualizar h_{max}, h_{min} $\mu = h_{max} - \textbf{\textit{rand()}} \cdot (h_{max} - h_{min})$
5. Escoger los nodos no seleccionados con heurística superior a μ y meterlo en *LRC*;
6. Escoger aleatoriamente un elemento s_{i*} de *LRC*
7. Incluir s_{i*} en *Sel*: *Sel* = *Sel* ∪ {s_{i*}}
8. Eliminar s_{i*} de *S*: *S* = *S* - {s_{i*}}

end while

Algoritmo de Enfriamiento Simulado

Procedimiento Simulated Annealing *(para maximizar)*

```
Start
  T ←T0;   s ← GENERATE();   Best Solution ← s;
  Repeat
       For cont = 1 to L(T) do /* Inner loop
         Start
       S' ←  NEIGHBORHOOD_OP(s); /* A single move
         Δf=f(s)–f(s');
         If ((Δf<0) or (U(0,1) <= exp(-Δf/T))) then
           S ← s';
                    If COST(s) is better than COST(Best Solution)
            then Best Solution ←  s;
         End
      T ← g(T); /* Cooling scheme. The classical one is geometric: T ← α·T
  until (Criterio de Parada); /* Outer loop
  Return(Best Solution);
End
```

Algoritmo de Enfriamiento Simulado

Procedimiento Simulated Annealing *(para maximizar)*

```
Start
  T ←T0;   s ← GENERATE();   Best Solution ← s;
  Repeat
      NExitos = 0; NVecinos=0;
       WHILE (Nexitos < MaxExitos && Nvecinos < MaxVecinos && NumEval < MaxEval)
        S' ←  NEIGHBORHOOD_OP(s); /* A single move
        Δf=f(s)–f(s');
        NVecinos ++; NumEval++;
        If ((Δf<0) or (U(0,1) <= exp(-Δf/T))) then
          S ← s';
          NExitos++;
           If COST(s) is better than COST(Best Solution) then Best Solution ←  s;
        End
      T ← g(T); /* Cooling scheme. The classical one is geometric: T ← α·T
  until (Criterio de Parada); /* Outer loop
  Return(Best Solution);
End
```

Enfriamiento Simulado para el MDP

- Exploración del vecindario: En cada iteración del bucle interno se genera una única solución vecina, **con el criterio de vecino de la práctica 1**, y se compara con la actual. **Se usa la factorización para el cálculo del coste**

- Esquema de enfriamiento: esquema de Cauchy modificado

- Condición de enfriamiento *L(T)*: cuando se genere un número máximo de soluciones vecinas, *máx_vecinos*, o se acepte un número máximo de los vecinos generados, *máx_éxitos*

- Condición de parada: cuando se alcance un número máximo de iteraciones o el número de éxitos en el enfriamiento actual sea 0

Enfriamiento Simulado para el SNIMP

- Exploración del vecindario: En cada iteración del bucle interno se genera una única solución vecina, **de forma aleatoria**, y se compara con la actual. **Se usa el mismo operador de mutación que la BL.**

- Esquema de enfriamiento: esquema de Cauchy modificado

- Condición de enfriamiento *L(T)*: cuando se genere un número máximo de soluciones vecinas, *máx_vecinos*, o se acepte un número máximo de los vecinos generados, *máx_éxitos*

- Condición de parada: cuando se alcance un número máximo de iteraciones o el número de éxitos en el enfriamiento actual sea 0