Material de apoyo Teórica XIII

Temario

Resolución de problemas de Programación Lineal Entera

- Métodos de resolución aproximada de problemas de PLE:
  - Heurísticas

Resolución aproximada de problemas de Programación Lineal Entera (Heurísticas)

Como dijimos en clases anteriores, los problemas combinatorios son problemas que tienen muchas combinaciones que pueden ser solución óptima del problema).

En este tipo de problemas el inconveniente es la Complejidad computacional que hace que tratar de resolverlo exactamente lleve una cantidad de tiempo superior al que podemos esperar.

Hoy en día las computadoras resuelven problemas mediante algoritmos que tienen como máximo una complejidad o costo computacional polinómico, es decir, la relación entre el tamaño del problema y su tiempo de ejecución es polinómica. Éstos son problemas agrupados en la clase P.

Los problemas que no pueden ser resueltos por nuestras computadoras (las cuales son Máquinas Determinísticas), que en general poseen costos factoriales o combinatorios pero que podrían ser procesados por una máquina no-determinista, están agrupados en la clase NP.

Una máquina determinística (como una computadora actual) no puede resolver estos problemas en un tiempo razonable.

Por eso, por lo general no se resuelven de manera exacta sino que se resuelven usando Algoritmos heurísticos (Heurísticas)

Algunos problemas de optimización combinatoria

- Problema de asignación (ver clase por página web)
- Problema del viajante de comercio (ver teórica)
- Problemas de cobertura de conjuntos (ver clase por página web)
- Planificación o Secuenciamiento de tareas
- Minimización de desperdicios en corte de material (como el 2.19)
- Problema de la mochila (como el 7.2)
- Problema de flujo máximo en redes
- Ruteo de vehículos (derivado del viajante) o de redes
- Determinación de caminos mínimos en grafos
- Asignación de recursos y horarios en instituciones educativas
- …

¿Cómo se resuelve un problema de optimización combinatoria?

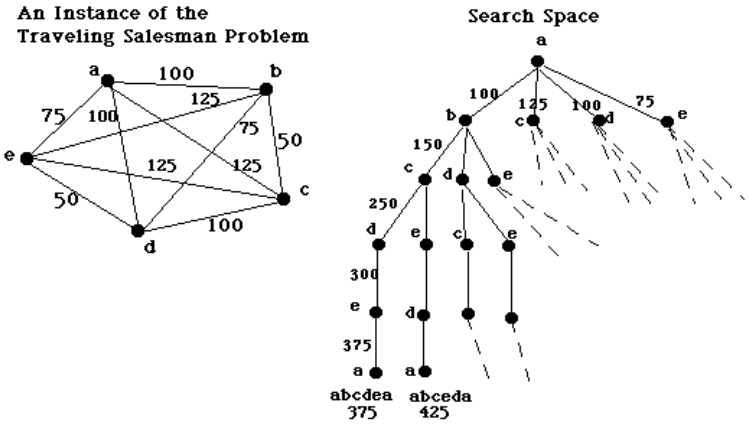
Se pueden resolver usando:

- Enumeración (fuerza bruta)
- Soluciones exactas
- Heurísticas

Si queremos resolverlo de manera exacta, vamos a necesitar demasiado tiempo (son problemas con muchas soluciones). Ya vimos en la teórica anterior la cantidad de operaciones que hay que hacer para resolver un problema de variables enteras.
Entonces, por lo general no se resuelven de manera exacta.

Por ejemplo:

Soluciones del TSP

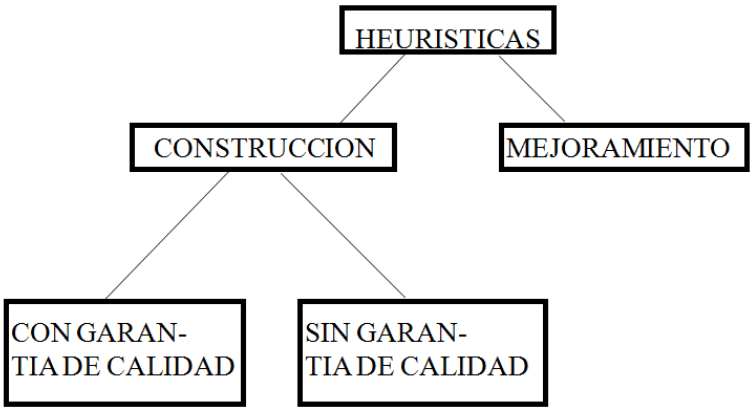


Como vemos, el espacio de búsqueda crece exponencialmente.

Cuándo usar heurísticas

- No existe un método exacto de resolución o requiere demasiado tiempo o recursos.
- No es necesario obtener la solución óptima (por ejemplo, porque no representa una gran ventaja sobre una subóptima)
- Datos poco fiables
- Limitaciones de tiempo (ej: se necesita responder en tiempo real)
- Paso intermedio en la aplicación de otro algoritmo

Clasificaciones de Heurísticas:



GARANTIA DE CALIDAD

Establece una relación entre la solución que se puede obtener con la heurística y la solución optima del problema.

En un problema de mínimo tendríamos que:

Valor obtenido por la heurística <= Constante * Valor Optimo

Siendo Constante >= 1

Problemas de secuenciamiento de tareas

Problemas con costo de "set-up"

- Se dispone de una máquina para realizar trabajos que se pueden numerar de 1 a k
- Si se realiza el trabajo j después del i hay que esperar un tiempo p_{ij} hasta que la máquina pueda empezar a procesar el trabajo j.
- Una vez que se termina con el último trabajo se empieza a procesar otra vez el primero.

EJEMPLOS:

- Máquina para producir pinturas
- Máquina para producir helados

Este problema en realidad es un viajante, porque es como si la máquina "viajara" por todas las tareas. Es un problema del viajante porque hay que determinar el orden (no está prefijado) y el valor del funcional (hay que minimizar el tiempo total) depende del orden. Los Pij son los "costos" Cij del problema del viajante.

Cuando en próximas clases veamos heurísticas para el problema del viajante podremos resolver este caso.

El problema de secuenciamiento con máquinas distintas

- Una serie de trabajos $J_1,..., J_k$ a ser procesados
- Cada trabajo J_i está compuesto de tareas u operaciones que se llaman $J_{i1},..., J_{im}$
- La operación o tarea J_{i1} del trabajo i debe ser procesada en la máquina 1, etc
- Para cada trabajo hay un orden en el cual se realizan las tareas u operaciones
- Cada operación o tarea J_{ij} tiene un tiempo de procesamiento p_{ij}.

Objetivo:

Decidir la secuencia en la cual se van a procesar las tareas integrantes de cada trabajo en las máquinas, de tal forma que se minimice el tiempo total.

- La secuencia que deben seguir las tareas dentro del trabajo es la misma para todos los trabajos:

"Flow shop scheduling"

- La secuencia varía con el trabajo:

"Job-shop scheduling"

Puede haber también tiempo de espera de trabajo (r_i: release time) y tiempo de entrega del trabajo (d_i: due time)

HEURISTICAS . Técnicas de despacho:

SPT: Elegimos la operación con menor tiempo de procesamiento

FCFS: Elegimos la operación que está lista desde hace más tiempo.

MWKR: Elegimos la operación asociada al trabajo al cual le queda más tiempo de procesamiento

MOPNR: Elegimos la operación que tiene el mayor número de operaciones que le siguen

LWKR: Elegimos la operación asociada al trabajo al cual le queda la menor cantidad de tiempo de procesamiento.

El problema de armar la bicicleta (un ejemplo de un problema de secuenciamiento de tareas)

¿Cómo mejorar el tiempo de fabricación de una bicicleta?. Las ventas andan bien

Se forman equipos de dos personas para armarlas a mano, porque el equipo de ensamblado no llega a satisfacer la demanda

Contratan a la empresa de Investigación Operativa **Modelos I** para que los ayude a resolver el problema-

La pregunta es: ¿Cuál es el menor tiempo necesario para que un equipo de 2 ensambladores arme una bicicleta

El ensamblado de una bicicleta se divide en tareas:

- A) Preparación del marco (7 minutos)
- B) Montar la rueda delantera (7 minutos)
- C) Montar la rueda trasera (7 minutos)
- D) Montar el sistema de frenos (2 minutos)
- E) Montar los piñones de cambios (3 minutos)
- F) Montar la cadena al piñón (2 minutos)
- G) Montar el piñón a la bicicleta (2 minutos)
- H) Montar el pedal derecho (8 minutos)
- I) Montar el pedal izquierdo (8 minutos)
- J) Ajustes finales (18 minutos)

¡No se puede armar la bicicleta en cualquier orden!

Restricciones de orden:

- A precede a B
- B, C, D, E y G preceden a J
- D precede a E y a F
- F y G preceden a H e I
- F precede a G
- D, E y A preceden a C

Restricciones de trabajo:

- Ningún ensamblador puede estar sin hacer nada si hay alguna tarea que se puede comenzar
- Todo ensamblador termina la tarea que empieza antes de pasar a otra tarea.

Este problema lo podríamos resolver por Programación Lineal Entera (parecido al 3.34), con variables que nos digan:

* En qué minuto comienza cada tarea

* Qué operario hace la tarea (bivalentes, una por cada operario para la misma tarea)

Y restricciones que controlen:

* Que antes de comenzar una tarea estén finalizadas las tareas precedentes

* Que no se hagan dos tareas al mismo tiempo

* Que al menos uno de los dos operarios haga la tarea

Algoritmo heurístico de resolución

¿Cuáles son las decisiones que tenemos que tomar en este caso?

- Cuando hay más de una tarea para hacer ¿cuál hacemos?
- Cuando hay más de un operario libre ¿cuál hace la tarea?

En el caso de la segunda decisión decimos que si hay dos operarios libres siempre

el primero es aquél al cual le damos la tarea (esa regla heurística la vamos a mantener en todas las heurísticas que hagamos). En el caso de qué tarea hacer cuando hay más de una disponible, la regla va a cambiar con cada algoritmo.

Reglas heurísticas (que se agregan a las restricciones de trabajo):

* El criterio para elegir un operario para realizar una tarea es el siguiente: si los dos operarios están desocupados, se le asigna tarea al operario 1 antes que al 2.

* El criterio para elegir una tarea a hacer (si tiene las precedencias cumplidas) es el siguiente: Si hay más de una tarea que se puede hacer, se hará primero la de mayor duración. Si hay más de una que tenga la mayor duración se elige entre ellas por orden alfabético.

Nota: Como criterio de selección se eligió la de mayor duración pero se podría elegir otro criterio ¿cuál podría ser?

1) Se asigna al operario 1 una tarea que no tenga precedencias
Si hay más de una *se elige la de mayor duración*
Si hay más de una se elige entre ellas por orden alfabético
2) Se asigna al operario 2 una tarea que no tenga precedencias
Si no hay ninguna, espera a que termine el operario 1
Si hay más de una *se elige la de mayor duración*
Si hay más de una se elige entre ellas por orden alfabético
3) Mientras queden tareas sin asignar
Cuando termina de trabajar un operario se le asigna una tarea que tenga las precedencias cumplidas.
Si no hay ninguna, espera a que termine el otro operario
Si hay más de una se elige la de mayor duración; si hay más de una se elige entre ellas por orden alfabético
Fin mientras
4) Se calcula el tiempo total que se tardó. FIN

Para resolver con el algoritmo planteado podemos hacer un diagrama de Gantt en el cual en el eje de abscisas se pone el tiempo y en el eje de ordenadas se pone una posición por operario, para ver en cuánto tiempo terminamos.

Vamos a plantear otro algoritmo heurístico de resolución: Reemplazamos en el que planteamos antes la palabra "mayor" por la palabra "menor". Ahora tenemos un algoritmo diferente. Resolvamos el problema con ese algoritmo para ver si tardamos menos tiempo que antes. Vamos a ver que tardamos más.

1) Se asigna al operario 1 una tarea que no tenga precedencias
Si hay más de una *se elige la de menor duración*
Si hay más de una se elige entre ellas por orden alfabético
2) Se asigna al operario 2 una tarea que no tenga precedencias
Si no hay ninguna, espera a que termine el operario 1
Si hay más de una *se elige la de menor duración*
Si hay más de una se elige entre ellas por orden alfabético
3) Mientras queden tareas sin asignar
Cuando termina de trabajar un operario se le asigna una tarea que tenga las precedencias cumplidas.
Si no hay ninguna, espera a que termine el otro operario
Si hay más de una se elige la de menor duración; si hay más de una se elige entre ellas por orden alfabético
Fin mientras
4) Se calcula el tiempo total que se tardó. FIN

¡Da peor!

¡¡¡El resultado depende de los datos planteados!!!!

* ¿Qué pasaría si el equipo que ensambla fuera de 3 personas?

* Adaptemos el algoritmo planteado anteriormente para tres personas y resolvamos el problema (esta tarea se las dejamos a ustedes)

* ¿Da mejor o peor? ¿Podría dar peor? SI, porque en las reglas de trabajo obligamos a que si alguien tiene trabajo para hacer debe hacerlo y a veces conviene esperar...

Cuando resolvemos con tres personas, da peor que con dos personas, justamente porque trata de "mantener ocupados" a los operarios y entonces los pone a hacer tareas que no son tan importantes, cuando las empezaron a hacer no las pueden interrumpir y las tareas importantes quedan para el final.