01/12/2021

Optimización y metaheurísticas

GRASP: Máquinas y Operaciones

Integrantes:

Juan Pablo Enriquez Pedroza 0228903

Ulises Gallardo Rodríguez 0229261

Sara Carolina Gómez Delgado 0226594

Luis Eduardo Robles Jiménez 0224969

5to semestre

Ing. Inteligencia Artificial

Profesor: Jonás Velasco Álvarez

Planteamiento del problema

En una industria se deben cumplir *m* cantidad de operaciones, para ello se cuenta con *n* máquinas, las cuales procesan las operaciones en un tiempo determinado, considerando además el tiempo de ajuste necesario que hace un operador para realizar dos operaciones diferentes.

El objetivo del problema es minimizar el tiempo máximo de entre todas las máquinas, al procesar sus respectivas operaciones más el tiempo de ajuste.

Técnicas metaheurística: GRASP

La metodología GRASP es un procedimiento iterativo que consiste en tres fases: construcción, mejora y actualización.

La fase constructiva (Greedy Randomized Adaptive) consiste en generar una buena solución factible a través de una función greedy, pero en lugar de escoger los mejores candidatos de dicha función, se opta por elegir uno de los mejores elementos al azar, haciendo uso de una lista restringida de candidatos (RCL). Esta lista puede estar restringida en cuanto a la cantidad de elementos (cardinalidad) o en cuanto a la calidad de los elementos.

Una vez incorporado el elemento a la solución parcial, se actualiza la lista de candidatos y se repite el procedimiento hasta que la solución sea construida.

Finalizada la fase anterior, se realiza una búsqueda local con el objetivo de mejorar la solución recientemente encontrada. Por último se actualiza la mejor solución global y se repite las 3 fases *k* números de iteraciones.

Función Greedy

Una función greedy es una estrategia de búsqueda que consiste en elegir la opción óptima en cada paso local.

Para el caso de la máquinas, se realizará el siguiente algoritmo greedy:

1. Se crea una matriz auxiliar donde cada casilla será el promedio de los tiempos de ajuste más el promedio de los tiempos de procesamientos de cada operación .
2. Al inicio del algoritmo, se parte de la operación con el menor tiempo de procesamiento. En las siguientes iteraciones, se define la siguiente operación de forma aleatoria a partir de uno aleatorio de sus mejores 3 vecinos en la mejor matriz (considerando promedio de operación y de tiempo de ajuste).

Después se asigna la siguiente operación que se definió antes, a la máquina donde afecte menos el resultado.

Búsqueda Local: Búsqueda tabú

La Búsqueda Tabú es una estrategia metaheurística que ayuda a resolver problemas de optimización combinatoria. Utiliza la búsqueda local para moverse entre vecindades y hace uso de una memoria temporal llamada “lista tabú”, la cual permite explorar nuevas soluciones (no siempre mejores) que evitan estancarse en óptimos locales o en soluciones recientemente generadas.

Elementos de la lista tabú:

I. Estado tabú: se clasifican ciertos movimientos o atributos de las soluciones como prohibidos (tabú)

II. Tenor del Tabú: número máximo de iteraciones en las que permanecerá una solución recientemente agregada.

III. Criterio de aspiraciones: Criterios que pueden modificar el estado de tabú de una solución, permitiendo admitir mejores soluciones que de lo contrario no serían visitadas.

Un ejemplo clásico de cómo podría implementarse la búsqueda tabú es en el problema de la mochila, donde a partir de una solución inicial se escoge el mejor candidato u atributo x donde $x \in X$ (solución inicial) y se agrega a la lista tabú, dicho elemento se bloqueará durante el tenor del tabú y después se liberará para dar paso a nuevas soluciones, realizando el procedimiento anterior en cada una de las iteraciones.

Problema de las máquinas

Para el problema de las máquinas y las operaciones, si cada operación se le puede asignar a cualquiera de las máquinas, entonces la complejidad del tiempo para calcular número de combinaciones sería O(n^m) donde n es el número de máquinas y m el número de operaciones totales, por lo que una complejidad exponencial no es muy viable cuando n y m son valores relativamente grandes.

Sin embargo, gracias a la función greedy podemos partir de una solución para después mejorarla con la búsqueda tabú en un tiempo reducido. No obstante es importante primero definir lo siguiente:

Representación de la solución

Se hace uso de una lista de *n* vectores que representan el número de máquinas. Cada vector contendrá *i* elementos que representan los índices de las operaciones que realizará cada máquina.

Evaluación de la solución

Su utiliza una función recibe una lista que calcula los tiempos de cada máquina y devuelve un vector con dichos tiempos, desde el cual podemos obtener los datos que usaremos a lo largo del programa (índice y valor de las máquinas con mayor y menor tiempo). El cálculo de los tiempos se hace recorriendo elemento por elemento cada máquina de la lista, sumando tanto sus tiempos de ajuste como de procesamiento.

Creación o perturbación de nuevas soluciones

Se obtendrá un nuevo vecindario modificando las máquinas con el mayor y el menor tiempo de procesamiento y ajuste acumulado, esta modificación se describe como sigue:

1. Se toma la *i*-ésima operación de la máquina con **mayor tiempo**.
2. Se inserta dicha operación en todas las posiciones posibles de la máquina con **menor tiempo** y se calcula el tiempo de procesamiento de cada iteración.
3. Se realizan los pasos anteriores *i* número de iteraciones, donde *i* es el número total de operaciones asignadas a una máquina.
4. Se obtiene el mejor tiempo del vecindario generado.
5. Se modifica la solución actual con la solución obtenida, transfiriendo la operación de la máquina con mayor tiempo a la de menor tiempo.
6. La operación transferida se agrega a la lista tabú.

Lista Tabú

Bloquea las operaciones que han sido modificadas recientemente durante el número de iteraciones determinado por el tenor tabú. Es decir, las operaciones contenidas en la lista tabú no podrán ser transferidas a otra máquina durante un tiempo determinado, evitando así que se pueda regresar a un estado anterior y repetir la solución.

Código

Código del proyecto

Resultados

# Instancia	Promedio	Desviación Estándar	Mínimo	Máximo	Referencia	Referencia Óptima
Instancia 1	98.8	6.097842	90	113	87	Optima
Instancia 2	86.7	5.33	82	102	82	Optima
Instancia 3	92.32	7.37	79	113	73	Factible
Instancia 4	85.86	5.45	72	98	72	Factible
Instancia 5	126.32	8.35	113	149	106	Factible
Instancia 6	129.66	9.8	116	168	106	Factible
Instancia 7	222.48	13.16	197	251	178	Factible
Instancia 8	210.46	10.64	192	236	179	Factible
Instancia 9	199.86	9.12	182	224	162	Factible
Instancia 10	192.16	9.77	179	214	156	Factible

Referencias

https://ccc.inaoep.mx/~emorales/Cursos/Busqueda/node66.html