Métodos Constructivos y Aleatorizados

Daniela Niño

Descripción de Algoritmos

Algoritmo 1 (constructivo)

En cada iteración de este algoritmo se selecciona el trabajo que cada máquina va a empezar a procesar en un momento determinado de tiempo. Es decir que cada iteración significa un momento en el tiempo.

Para la primera iteración, como ninguna máquina ha procesado algún trabajo, el tiempo inicial es cero (t=0).

Algoritmo 1 (constructivo)

El algoritmo para cada iteración hace lo siguiente:

1. Verifica que aún no se cumpla el criterio de parada
2. Verifica cuáles máquinas están disponibles en el momento determinado de tiempo de la iteración
3. Verifica cuáles trabajos están disponibles para cada máquina disponible en el momento determinado de tiempo de la iteración
4. Para cada máquina disponible selecciona entre los trabajos disponibles el que tenga **menor** tiempo de ejecución
5. Calcula el nuevo instante de tiempo para evaluar en la siguiente iteración

Algoritmo 1 (constructivo)

Explicación del paso a paso:

1. Verifica que aún no se cumpla el criterio de parada

El algoritmo para de iterar cuando ya todos los trabajos fueron procesados por todas las máquinas.

Algoritmo 1 (constructivo)

2. Verifica cuáles máquinas están disponibles en el momento determinado de tiempo de la iteración

El algoritmo busca cuáles máquinas no están procesando ningún trabajo para el instante de tiempo (t) de la iteración.

3. Verifica cuáles trabajos están disponibles para cada máquina disponible en el momento determinado de tiempo de la iteración

El algoritmo busca cuáles trabajos no están siendo procesados para el tiempo (t) de la iteración.

Algoritmo 1 (constructivo)

4. Para cada máquina disponible selecciona entre los trabajos disponibles el que tenga **menor** tiempo de ejecución

En este punto ya sabremos cuáles máquinas están disponible (las encontradas en el punto 2) y cuáles trabajos están disponibles (los encontrados en el punto 3), por lo tanto, conocemos cuáles trabajos pueden ser procesados por cada máquina en el tiempo (t) de esta iteración. Entonces se procede a elegir para cada máquina disponible, el trabajo disponible que tenga **menor tiempo de procesamiento**. Los trabajos elegidos serán los que cada máquina empezará a procesar en el tiempo actual (t).

Algoritmo 1 (constructivo)

5. Calcula el nuevo tiempo para evaluar la siguiente iteración

El algoritmo define cuál será el tiempo de la siguiente iteración. Para saber cual es el instante de tiempo que sigue para ser analizado en la siguiente iteración, se hace lo siguiente:

- Se necesita un **arreglo** que indique en qué momento del tiempo se libera cada máquina, es decir, el arreglo indica en qué momento de tiempo cada máquina va a terminar de procesar el trabajo que se encuentra procesando en el tiempo actual (t).
- El tiempo a seleccionar para ser evaluado en la siguiente iteración debe ser el valor menor que se encuentre en este **arreglo**, con la única condición es que este momento de tiempo a seleccionar debe ser mayor al tiempo actual (t).

Algoritmo 2 (ruido)

Este algoritmo funciona igual que el algoritmo 1, con la diferencia de que cuando se va a determinar cual trabajo va a ser procesado por cada máquina, se elige el tiempo de ejecución menor a partir de un **dataset con ruido**.

El **dataset con ruido** es la **matriz de tiempos de procesamiento** pero con la suma de un **ruido p**. El **ruido p** es un valor aleatorio entero que proviene de una distribución uniforme entre **[-r, r]**, donde **r** es un hiperparámetro de tipo entero.

Algoritmo 2 (ruido)

Es decir que el tiempo de procesamiento con ruido se define como:

$$t_{jk}^{ruido} = t_{jk} + \rho_{jk}$$

donde t_{jk} es el tiempo de procesamiento del trabajo j en la máquina k

donde ϱ_{jk} es el ruido agregado al tiempo de procesamiento del trabajo j en la máquina k

Algoritmo 3 (GRASP)

Este algoritmo funciona igual que el algoritmo 1, con la diferencia de que en vez de elegir el trabajo con menor tiempo de ejecución, busca los k trabajos con menor tiempo de ejecución y elige uno aleatoriamente.

En otras palabras, se elige para cada máquina disponible, un trabajo aleatorio entre los k trabajos disponibles que tenga menor tiempo de procesamiento.

Criterio de Selección

Criterio de selección de trabajos

Como fue explicado anteriormente, en cada iteración del algoritmo se escogen los trabajos con el tiempo de procesamiento **menor**. Para escoger este criterio de decisión se hizo una prueba con el algoritmo #1 (ya que este determinista) donde se escogieron tanto los trabajos con tiempo de procesamiento **menor** y **mayor**.

Los resultados de esta prueba fueron que para **todas** las instancias fue mejor la función objetivo cuando se tomaban los tiempos de procesamiento **menores**, por lo tanto, todos los algoritmos se hicieron usando este criterio.

A continuación se encuentran los resultados de la **función objetivo** para cada instancia por cada criterio de decisión (tiempo menor y mayor).

Criterio de selección de trabajos

Criterio	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
min	1462	1446	1865	1667	2175	1965	2335	2432
max	1701	1755	2046	2069	2266	2324	2417	2501

Criterio	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
min	2499	2710	3856	3266	3606	3639	6232	5973
max	2925	3140	3880	4028	3989	3959	7038	7170

* El valor subrayado en verde significa que la función objetivo fue la mejor para esa instancia.

Criterio de selección de trabajos

Criterio	Total
min	47128
max	53208

* Este valor indica la suma de la función objetivo de todas las instancias.

Comparación con Diferentes Valores de Parámetros

Comparación con diferentes valores de parámetros

El algoritmo ruido fue probado con r = 5, r = 10 y r = 15, se eligieron estas r porque cubren rangos pequeños y amplios de tiempo. Un r = 5 cubre un rango de 10 unidades de tiempo debido a que la distribución uniforme selecciona número del intervalo [-5, 5] y un r = 15 cubre un rango de 30 unidades de tiempo debido a que la distribución uniforme selecciona número del intervalo [-15, 15] .

El algoritmo GRASP se probó con k = 3, k = 4 y k = 5, solo se usaron números pequeños debido a que se hicieron varias pruebas y se descubrió en cada iteración hay muy pocos trabajos disponibles por máquina, por lo tanto si se pone un k muy grande, se aumenta mucho la aleatoriedad del algoritmo.

Comparación con diferentes valores de parámetros

Debido a que ruido y GRASP son aleatorios, cada uno de estos algoritmos se ejecutó tres veces por parámetro, y a partir de los tres resultados obtenidos se hizo un promedio de ellos para cada parámetro. Esto se hizo para que se tuviera más certeza de cuál valor de r y k es mejor, es decir, para que que no se elija el valor de los parámetros tan aleatoriamente.

Los valores mostrados en las siguientes tablas son los valores promedio calculados de cada algoritmo por cada valor de parámetro.

En este link se encuentran más detalles de los datos datos de **ruido** (ver las diferentes hojas del libro) y en este link se encuentran más detalles de los datos datos de **GRASP** (ver las diferentes hojas del libro).

Algoritmo 2 (ruido)

r	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
5	1460	1488	1825	1757	2177	1948	2355	2470
10	1502	1534	1840	1697	2199	2069	2307	2384
15	1454	1476	1830	1775	2163	1956	2317	2491

r	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
5	2597	2752	3437	3281	3528	3708	6192	6045
10	2640	2719	3398	3232	3624	3777	6445	6017
15	2534	2729	3582	3282	3728	3731	6215	6078

* El valor subrayado en verde significa que la función objetivo fue la mejor para esa instancia.

Algoritmo 2 (ruido)

r	Total
5	47021
10	47386
15	47340

* Este valor indica la suma de la función objetivo de todas las instancias.

Algoritmo 3 (GRASP)

k	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
3	1687	1605	1688	1845	2183	2092	2303	2328
4	1526	1581	1876	1700	2268	2085	2316	2468
5	1756	1556	1713	1796	2198	2160	2276	2405

k	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
3	2671	2638	3377	3284	3715	3745	6179	5898
4	2761	2788	3490	3381	3651	3758	6251	5942
5	2701	2904	3495	3361	3680	3674	6155	5911

* El valor subrayado en verde significa que la función objetivo fue la mejor para esa instancia.

Algoritmo 3 (GRASP)

k	Total
3	47238
4	47841
5	47741

* Este valor indica la suma de la función objetivo de todas las instancias.

Comparación con diferentes valores de parámetros

En general, para el algoritmo 2 (ruido) fue mejor r = 5, esto **probablemente** sucede ya que como el rango no es muy grande, no se afectan mucho los valores reales de los trabajos, pues con un k más grande ya el valor original cambia mucho y por lo tanto aumenta la aleatoriedad del algoritmo.

En general, para el algoritmo 3 (GRASP) fue mejor k = 3, esto **probablemente** sucede ya que como se mencionó anteriormente, para cada iteración hay muy pocos trabajos disponibles por máquina, entonces con un k muy grande, puede suceder que para varias iteraciones los trabajos disponibles sean menos que el valor k, y por lo tanto, el algoritmo selecciona entre **todos** los trabajos disponibles y entonces se aumenta la aleatoriedad del algoritmo.

Comparación entre Algoritmos

Comparación entre Algoritmos

En las siguientes tablas se encuentra una comparación entre la función objetivo de los tres algoritmos, para ruido se usa r = 5 y para GRASP k = 3. Adicionalmente, los valores mostrados para ruido y GRASP son los valores promedio calculados al ejecutar estos algoritmos tres veces.

En este link se encuentra más información de los resultados de las tres ejecuciones de ruido en la hoja llamada “r = 5” y en este link se encuentra más información de los resultados de las tres ejecuciones de GRASP en la hoja llamada “k = 3”.

Comparación entre Algoritmos

Algorit.	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
cons.	1462	1446	1865	1667	2175	1965	2335	2432
ruido	1460	1488	1825	1757	2177	1948	2355	2470
grasp	1687	1605	1688	1845	2183	2092	2303	2328

Algorit.	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
cons.	2499	2710	3856	3266	3606	3639	6232	5973
ruido	2597	2752	3437	3281	3528	3708	6192	6045
grasp	2671	2638	3377	3284	3715	3745	6179	5898

* El valor subrayado en verde significa que la función objetivo fue la mejor para esa instancia.

Comparación entre Algoritmos

Algoritmo	Total
constructivo	47128
ruido (r=5)	47021
grasp (k=3)	47238

* Este valor indica la suma de la función objetivo de todas las instancias.

Comparación entre Algoritmos

En general, el mejor algoritmo fue el 1 (ruido) debido a que la suma de la función objetivo de todas las instancias fue la menor, pero al mismo tiempo, si contamos la cantidad de veces que la función objetivo fue mejor por algoritmo los resultados son estos:

- Para el algoritmo 1, la función objetivo fue mejor para 6 instancias
- Para el algoritmo 2 (ruido), la función objetivo fue mejor para 4 instancias
- Para el algoritmo 3 (GRASP), la función objetivo fue mejor para 6 instancias

Por lo tanto, los mejores algoritmos serían los 1 y 2 (GRASP).

Comparación con una Cota Inferior

Comparación con una cota inferior

Para sacar las cotas inferior se hizo lo siguiente **por cada instancia**:

- Se calculó el tiempo mínimo que cada trabajo demora en ser procesado por todas las máquinas. Para cada trabajo j se hizo la siguiente operación:

$$\sum_{k=1}^{m} t_{jk}$$

donde m es el número total de máquinas

donde t_{jk} es el tiempo de procesamiento del trabajo j en la máquina k

Comparación con una cota inferior

- Se calculó el tiempo mínimo que cada máquina demora en procesar todos los trabajos. Para cada máquina k se hizo la siguiente operación:

$$\sum_{j=1}^{n} t_{jk}$$

donde n es el número total de trabajos

donde t_{jk} es el tiempo de procesamiento del trabajo j en la máquina k

Comparación con una cota inferior

- Se hizo un arreglo con el valor de todos los resultados calculados en los dos puntos anteriores, por lo tanto, quedó un arreglo de longitud n+m.
- Posteriormente, se buscó el **valor máximo del arreglo**, el cual es la cota inferior de la instancia, pues sabemos que el algoritmo no puede terminar antes que todos los trabajos sean procesados por todas las máquinas, por lo tanto no puede demorar menos que el valor máximo de ese arreglo.

Comparación con una cota inferior

Algorit.	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
cons.	1462	1446	1865	1667	2175	1965	2335	2432
ruido	1460	1488	1825	1757	2177	1948	2355	2470
grasp	1687	1605	1688	1845	2183	2092	2303	2328
cota	977	942	1139	1251	1217	1240	1764	1774

Comparación con una cota inferior

Algorit.	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
cons.	2499	2710	3856	3266	3606	3639	6232	5973
ruido	2597	2752	3437	3281	3528	3708	6192	6045
grasp	2671	2638	3377	3284	3715	3745	6179	5898
cota	1830	1761	2760	2756	2868	2848	5464	5181

Comparación con una cota inferior

Algoritmo	Total
constructivo	47128
ruido (r=5)	47021
grasp (k=3)	47238
cota	35772

* Este valor indica la suma de la función objetivo de todas las instancias.

Tiempo de Cómputo

Tiempo de cómputo

En las siguientes tablas se encuentra una comparación entre el tiempo de cómputo de los tres algoritmos. Los valores mostrados son los valores promedio calculados al ejecutar cada algoritmo tres veces, es decir que los valores en las tablas es el tiempo de cómputo promedio por algoritmo. Además, para ruido fue seleccionado r = 5 y para GRASP k = 3.

En este link se encuentra más información de los resultados de las tres ejecuciones del algoritmo 1, en este link se encuentra más información de los resultados de las tres ejecuciones del algoritmo 2 (ruido) en la hoja llamada “r = 5” y en este link se encuentra más información de los resultados de las tres ejecuciones de GRASP en la hoja llamada “k = 3”.

Tiempo de cómputo

Unidad de tiempo: segundos

Algorit.	JSSP1	JSSP2	JSSP3	JSSP4	JSSP5	JSSP6	JSSP7	JSSP8
cons.	0.0400	0.0438	0.0600	0.0517	0.1116	0.1089	0.0925	0.0884
ruido	0.0886	0.0711	0.0662	0.0677	0.1715	0.1535	0.1708	0.1268
grasp	0.0629	0.0605	0.0968	0.0862	0.2190	0.1920	0.1413	0.1282

Algorit.	JSSP9	JSSP10	JSSP11	JSSP12	JSSP13	JSSP14	JSSP15	JSSP16
cons.	0.1657	0.1745	0.1426	0.1587	0.2761	0.2695	0.6331	0.6952
ruido	0.2261	0.2251	0.2123	0.2110	0.3797	0.3994	0.8928	0.8742
grasp	0.2905	0.3311	0.2373	0.2546	0.4687	0.4734	1.1169	0.9966

* El valor subrayado en verde significa que el tiempo de cómputo fue la mejor para esa instancia.

Tiempo de cómputo

Unidad de tiempo: segundos

Algoritmo	Total
constructivo	3.1124
ruido (r=5)	4.3370
grasp (k=3)	5.1560

* Este valor indica la suma de la función objetivo de todas las instancias.

Tiempo de cómputo

Para todas las instancias siempre el algoritmo con el tiempo de cómputo más rápido fue el 1, esto sucede debido a que los otros dos algoritmos deben realizar pasos adicionales, los cuales hacen que demoren más tiempo. El algoritmo 2 debe agregar el ruido y el algoritmo 3 debe buscar k trabajos, en vez de solo uno.

Entre el algoritmo 2 y el 3, en general fue más rápido el 2 (ruido).

Conclusiones

Conclusiones

- El criterio de decisión de seleccionar los trabajos con menor tiempo de ejecución es significativamente mejor que seleccionar los trabajos con mayor tiempo de ejecución, pues como se mencionó anteriormente, para **todas** las instancias fue mejor cuando se seleccionaba el tiempo menor.

  Adicionalmente, se puede notar que en la diferencia entre ambos criterios es significativa ya que al seleccionar el tiempo de ejecución **menor**, la función objetivo indica que los trabajos fueron procesados en promedio 11% más rápido.

  Por esta razón, es importante hacer varias pruebas para poder seleccionar un criterio de selección de trabajos.

Conclusiones

- Para garantizar elegir el valor apropiado de los parámetros en los algoritmos aleatorios, es importante ejecutarlos varias veces y sacar un promedio. Esto dado que la aleatoriedad genera que en unas ocasiones el valor adecuado de los parámetros den peores resultados. Por ejemplo, para el algoritmo ruido, en una ocasión el k = 10 generó un mejor resultado que k = 5 (este siendo el mejor valor en promedio).
- Entre más grandes sean los parámetros r y k, más aleatorio se vuelve el algoritmo y por lo tanto empeora su función objetivo.

Conclusiones

- La diferencia entre los tres algoritmo no es significativa, pues los valores de la función objetivo no varían en un rango muy grande entre algoritmos.

  Adicionalmente, como los algoritmos de ruido y GRASP son aleatorios, puede haber ocasiones en las que el resultado de estos algoritmos sea mejor que el del algoritmo 1 y ocasiones en las que el resultados sea peor.

  Por lo tanto, no podemos definir cuál algoritmo es el mejor, pues en unas ocasiones es mejor la función objetivo en el algoritmo 1, en otras es mejor el algoritmo 2 y en otras el algoritmo 3.

Conclusiones

- El tiempo de cómputo del algoritmo 2 (ruido) probablemente es menor que el del algoritmo 3 (GRASP), debido a que el paso adicional del algoritmo 2 se hace una única vez (agregar ruido) mientras que el paso adicional del algoritmo 3 (GRASP) se debe hacer para cada iteración por cada máquina disponible (seleccionar k trabajos).

  Por lo tanto, probablemente el algoritmo ruido es más rápido que GRASP ya que debe hacer menos operaciones adicionales.

Referencias Bibliográficas

Referencias

Rivera Agudelo, J. C. (2023). Ruido [Diapositivas]. Escuela de Ciencias Aplicadas e Ingeniería, Universidad EAFIT. https://interactivavirtual.eafit.edu.co/

Rivera Agudelo, J. C. (2023). Construcciones GRASP [Diapositivas]. Escuela de Ciencias Aplicadas e Ingeniería, Universidad EAFIT. https://interactivavirtual.eafit.edu.co/