Aplicación de la búsqueda local iterada y búsqueda tabú para resolver el problema de programación de tareas

Alejandro Guzmán[1]

[1] Universidad de los Andes
Cra 1 Nº 18A - 12, 28922, Bogota, Colombia

Abstract

En este documento se presentará una solución para el problema de programación de tareas mediante la utilización de las heurísticas de búsqueda local iterada (ILS) y búsqueda tabu (TS). La heurística ILS fue escogida debido a que, según la literatura cuando el problema flow-shop es de gran escala encontrar óptimos locales y luego salir de estos escala la complejidad del problema, teniendo en cuenta esto se tuvo en cuenta una de las propiedades de la heurística en cuestión la cual es, encontrar buenas soluciones sin quedarse atascado en regiones subóptimas del espacio de búsqueda por otro lado, la búsqueda tabu se escogió debido a que esta nos sirve para crear estas iteraciones necesarias para llevar a cabo la ILS y por su bajo costo computacional teniendo en cuenta esto, se evaluará el rendimiento de estas, resolviendo las instancias más grandes definidas en Taillard 1993 [3], aparte se compararán los resultados con los de otras heurísticas y metaheurísticas para determinar si la aplicación de estas es efectiva o no.

1 Introducción

Actualmente el problema de programación de tareas representa un papel importante en los ámbitos de la producción, distribución, transporte, procesamiento de información y comunicaciones, considerado como un problema NP-Hard debido a que su solución no se encuentra en tiempo polinomial. Debido a su importancia, se han desarrollado varios algoritmos enfocados en minimizar el Makespan de la mejor programación de tareas y a su vez, reduciendo lo más posible la complejidad computacional [1]. Este problema fue introducido y formulado por Johnson 1994[2] el cual logro obtener la programación óptima para la instancia con dos y tres máquina, sin embargo la segunda en un caso restringido, después de esto se implementaron muchos algoritmos para solucionar el problema como decision tree algorithm (DT), scatter search algorithm, también se implementaron métodos heurísticos como: standard deviation heuristic , procedure of constructive heuristic entre muchas otras.

Independientemente de todo el trabajo realizado alrededor de los años para mejorar la resolución de este problema, en la actualidad sigue vigente un problema en concreto y este es el de encontrar el óptimo local cuando se resuelven problemas de gran magnitud. Teniendo en cuenta que la finalidad de todas las metodologías planteadas se enfocan en reducir el Makespan y que uno de los principales problemas de los algoritmos planteados es encontrar y luego escapar del óptimo local, se realizará la implementación de la búsqueda local iterada y la búsqueda tabu, utilizando instancias con un alto número de máquinas y usuarios, las cuales fueron planteadas por Taillard 1993 [3], específicamente se usarán las que se componen de 500 trabajos y 20 máquinas.

1.1. Descripción del problema

El problema del Flow-Shop o problema de secuenciación de tareas de producción lineal consiste en programar de forma óptima un conjunto de n trabajos en m máquinas, considerando que todas las tareas tienen el mismo orden de procesamiento, los tiempos de ejecución varían según la etapa que sea visitada, no se consideran tiempos de ajuste de las máquinas entre un trabajo y otro, los trabajos no pueden ser interrumpidos, minimizando el tiempo total requerido para terminar todas las tareas (makespan). Este tipo de problemas los podemos caracterizar mediante los siguientes puntos [4]:

- Cada trabajo j_k solo puede realizarse en cada máquina una solo vez, donde k=1,…n
- Cada máquina i_h solo puede procesar un trabajo a la vez donde h= 1,…, m

- Cada trabajo j tarda un tiempo en ser procesado en cada máquina i.
- El tiempo de procesamiento de cada trabajo cambia dependiendo de la máquina ya que cada trabajo se ejecutará en la máquina 1 a m ordenadamente.
- El tiempo de procesamiento de cada trabajo incluye el tiempo de preparación más el tiempo de funcionamiento de una máquina.
- Se supone que la hora de inicio es cero
- Cada trabajo tiene un tiempo de finalización c para terminar su procesamiento en una máquina que toma dos parámetros $c(j_k i_h)$ como es el tiempo de finalización del trabajo j en una máquina i.

2. Marco teórico

2.1 Heurística de búsqueda local iterada

La metaheurística de ILS se basa en una estrategia de búsqueda local utilizando una única solución a lo largo del proceso iterativo. ILS es un método de bajo tiempo computacional utilizado para mejorar la calidad de los óptimos locales, que es útil en problemas de alta dimensión. Generalmente, el proceso realizado por el método ILS comienza con una solución inicial aleatoria (codificación binaria) dentro del espacio de búsqueda predefinido, y el criterio de parada puede ser un número de iteraciones. La idea fundamental del ILS se rige por los pasos de búsqueda local, perturbación y el criterio de parada [5].

2.2 Búsqueda tabú

Búsqueda tabú es una heurística que se utiliza para resolver problemas de optimización. El rendimiento de TS se ha mejorado con respecto a la técnica de búsqueda local escapando de las soluciones ya visitadas del problema y sus vecinos. Esto detiene la búsqueda de soluciones óptimas locales, esta crea candidatos que son básicamente un conjunto de varias variables enteras. Estos candidatos varían en uno o más bits de la solución óptima actual y se descartan de la lista creada. Luego, los subproblemas del problema original se resuelven para cada candidato mediante la técnica basada en gradientes. Para crear la semilla para la próxima generación, se selecciona el candidato presente con el mejor valor objetivo [6].

3. Metodología

3.1 Formulación matemática del problema

Los problemas Flow-Shop se caracterizan por estar compuestos de una serie de trabajos *j* y máquinas *i*, donde $(1 \leq i \leq m, 1 \leq j \leq n)$ a su vez, cada trabajo tiene asociado un tiempo de procesamiento en cada máquina d_{ij}.

Para el problema es importante calcular el límite inferior del makespan y este se calcula de la siguiente manera. Sea b_i el tiempo mínimo antes de que la máquina *i* comience a trabajar y a_i el tiempo mínimo que la máquina se mantiene inactiva después de que su trabajo termina hasta el final de las operaciones y T_i es el tiempo de procesamiento total. Teniendo en cuenta esto se tiene que:

$$b_i = \min_j \sum_{k=1}^{i-1} d_{kj} \quad (1)$$

$$a_i = \min_j \sum_{k=i+1}^{m} d_{kj} \quad (2)$$

$$T_i = \min_j \sum_{k=i+1}^{m} d_{ij} \qquad (3)$$

En relación con el makespan óptimo *C* se tiene que:

$$LB = \max_i \left((b_i + T_i + a_i), \max_j \left(\sum_{k=i+1}^{m} d_{ij} \right) \right) \leq C \quad (4)$$

3.2 Algoritmo

El algoritmo aplicado para solucionar el Flow-Shop se construyó en Python, apoyándonos en el repositorio [7]. Para la resolución del problema se utilizó la combinación de dos heurísticas las cuales son, Tabú Search e Iterated Local Search, la primera se utilizó para obtener mejoras de la solución inicial del algoritmo y la segunda para realizar el proceso iterativo de analizar las vecindades mediante la perturbación.

1. Se formula el problema como un problema de programación de tareas, con un conjunto de trabajos que deben completarse, un conjunto de máquinas que se pueden usar para completar los trabajos y los tiempos de procesamiento para cada trabajo en cada máquina.
2. Crear una solución inicial para el problema en este caso se decidió que esta debía ser por enumeración es decir que el trabajo 1 va primero, luego el trabajo 2 y así sucesivamente.
3. Aplicar el algoritmo de búsqueda tabú, a la solución inicial para mejorar su calidad esta se diseña de tal manera que explore la vecindad de la solución actual y encuentre mejores soluciones haciendo pequeños cambios en la solución actual.
4. Paralelamente, en cada iteración aplicar la perturbación a cada mejora que encuentre la búsqueda tabú, y luego aplicarle la búsqueda tabú a esta.
5. Comparar los costos de la configuración original con los de la configuración generada por la perturbación y guardar la mejor configuración.
6. Volver al paso 3 e iterar hasta que el criterio se cumpla, ya sea un número de iteraciones o hasta que se esté suficientemente cerca de una solución específica.

La figura 1, describe en pseudocódigo el funcionamiento del algoritmo.

```
ALGORITHM flow_shop_ils(jobs, machines)
    BEGIN ALGORITHM
        iterations <- n
        iteration <- 0
        initial_solution <- create_initial_solution(jobs, machines)
        current_solution <- initial_solution
        best_solution <- initial_solution
        FOR each iteration until iteration <= iteratons
            improved_solution <- apply_Tabu_search(current_solution)
            disturbed_solution <- apply_disturbance(improved_solution)
            current_solution <- disturbed_solution
            best_solution <- compare_and_save_best(best_solution, current_solution)
            iteration += 1
        END FOR
        RETURN best_solution
    END ALGORITHM
```

Figura 1 Algoritmo en pseudocódigo

Añadido a esto, el algoritmo se puede dividir con esto se quiere decir que este también puede resolver el problema únicamente utilizando la Búsqueda Tabu, la diferencia con el inicial es que elimina el paso de la perturbación y por ende el de la comparación, de resto

mantiene la misma estructura.

4. Resultados

Al algoritmo del Búsqueda Local Iterada (ILS) se le ingresaron las 10 instancias previamente enunciadas, y se corrieron 10 iteraciones ya que, se hicieron corridas del algoritmo para la instancia 1, con cinco, 10, 15 y 20 iteraciones y los valores de la solución son muy similares sin embargo, el tiempo computacional a más iteraciones aumenta considerablemente por lo tanto, se decidió correr 10 iteraciones para todas las instancias obteniendo los resultados reportados en la tabla 1.

Tabla 1 Resultados para las 10 instancias del Flow Shop mediante la ILS

Instance	id	BKS	ILS Solution	GAP	CPU Time ILS
tai 500_20_1	1	26040	28990	11.33%	152.848130941391
tai 500_20_2	2	26520	29466	11.11%	152.3644871711731
tai 500_20_3	3	26371	29242	10.89%	153.656968832016
tai 500_20_4	4	26454	29286	10.71%	151.86301708221436
tai 500_20_5	5	26334	29070	10.39%	153.4178650379181
tai 500_20_6	6	26477	29379	10.96%	153.15439295768738
tai 500_20_7	7	26389	29053	10.10%	155.9342019557953
tai 500_20_8	8	26560	29188	9.89%	151.43218898773193
tai 500_20_9	9	26005	29098	11.89%	160.229749917984
tai 500_20_10	10	26457	29027	9.71%	155.08094000816345

Para el algoritmo de Búsqueda Tabu por sí solo, se hicieron experimentos con 50, 100 y 200 iteraciones sin embargo, las diferencias entre las soluciones eras mínimas, por lo cual se decidió correr el algoritmo con 50 iteraciones, por optimalidad computacional de esta manera se obtuvieron los resultados presentados en la tabla 2.

Tabla 2 Resultados para las 10 instancias del Flow Shop mediante la TS

Instance	id	BKS	TS Solution	GAP	CPU Time TS
tai 500_20_1	1	26040	28763	10.46%	69.07873201370239
tai 500_20_2	2	26520	29359	10.71%	69.72688293457031
tai 500_20_3	3	26371	29032	10.09%	68.8897979259491
tai 500_20_4	4	26454	28925	9.34%	68.45523309707642
tai 500_20_5	5	26334	28949	9.93%	72.22865009307861
tai 500_20_6	6	26477	29374	10.94%	71.86936593055725
tai 500_20_7	7	26389	28910	9.55%	69.55606484413147
tai 500_20_8	8	26560	29018	9.25%	69.54661393165588
tai 500_20_9	9	26005	28882	11.06%	69.63009786605835
tai 500_20_10	10	26457	28811	8.90%	69.90780425071716

Como se puede observar en las tablas anteriores, si analizamos los GAP entre las BKS y las soluciones de cada heurística para cada instancia, se puede determinar que la aplicación de la Búsqueda Tabú por si sola arroja mejores resultados que la ILS, ya que para la primera el GAP es en promedio de 10.7% en cambio para la segunda es de 10.02%, esto se debe a que la TS al ser una búsqueda menos costosa se nos da la posibilidad de correr más iteraciones y de esta manera nos acercarnos más al óptimo.

5. Discusión

Habiendo hecho el análisis de las brechas con las mejores soluciones conocidas para el problema, se decidió comparar las soluciones encontradas, con resultados presentados en la literatura, para terminar de analizar si nuestra implementación era de calidad por lo tanto, se realizó la comparación de la solución de las heurísticas planteadas, con soluciones encontradas a partir de la aplicación de: PSO, una combinación entre GRASP, Genetic algorithm y VNS (NEGAVNS), una nueva estrategia de aprendizaje de enjambre de partículas introducido en el algoritmo RDPSO para guiar la búsqueda con el fin de encontrar las mejores soluciones personales y globales [7]. Cabe resaltar estas heurísticas se corrieron con 30 iteraciones de cada instancia, de esta manera en la tabla 2 se presenta el promedio de nuestras soluciones y el promedio de las soluciones arrojadas por estos algoritmos.

Tabla 3 Promedio de las soluciones de cada metodología

	ILS	TS	PSO	NEGAVNS	RDPSO
Solution	29189	29061	26737	26228	26656

Tabla 4 GAP entre el promedio de las soluciones de la ILS con las otras metodologías

	PSO	NEGAVNS	RDPSO
GAP ILS	9.2%	11.3%	9.5%

Tabla 5 GAP entre el promedio de las soluciones de la TS con las otras metodologías

	PSO	NEGAVNS	RDPSO
GAP TS	8.7%	10.8%	9.0%

Como se pudo observar en comparación con otros métodos heurísticos nuestras soluciones siguen siendo mayores, inclusive en el caso del TS para la cual se corrieron 50 iteraciones sin embargo, hay que contemplar que en el apartado anterior en el cual calculamos el GAP con las mejores soluciones conocidas, estas fueron calculadas sin considerar el tiempo computacional, en el mismo estudio donde se exponen estas se dice explícitamente que no es un parámetro importante y en la comparación con las heurísticas, estas se corrieron con un tiempo límite de hasta un día, por lo cual es lógico que con esas capacidades computacionales se puedan llegar a mejores soluciones, de igual forma nuestras metodologías llegan a soluciones bastante buenas con GAP´S que en promedio rondan los 10%, por lo tanto si corriéramos durante un día nuestra búsqueda Tabu o nuestro ILS es muy probable que se encuentren soluciones mucho mejores a las expuestas.

6. Conclusiones

- El número de iteraciones, tanto para la Iterative Local Search como pata la Tabu Search, infiere mucho en el resultado final del ordenamiento y su calidad.
- La heurística ILS propuesta, en problemas grandes tiene un costo computacional demasiado alto si se usan bastantes iteraciones en función de encontrar una solución cercana al óptimo, sin embargo es una heurística más robusta que la TS.
- El TS planteado brinda soluciones buenas en tiempos computacionales muy buenos sin embargo, en problemas tan grandes nos arriesgamos a quedar atrapados en óptimos locales y caer en el error de no explorar otras vecindades del espacio.
- A comparación de metodologías más robustas el GAP del TS con estas no es tan alto y los costos de esta son bajos a comparación, por lo cual se puede decir que en casos de necesitar soluciones rápidas, es una muy buena opción aplicarla.
- Para pocas iteraciones la perturbación usada en la ILS no tienen ningún efecto.

- El costo computacional de la búsqueda local iterada para obtener soluciones cercanas al óptimo son altas.

Por último, se puede decir que la búsqueda local iterada produce una solución aceptable para el problema Flow-Shop, sin embargo por su costo computacional sería recomendable explorar otras opciones, en cuando a la Búsqueda Tabú, esta tiene una gran relación calidad-costo, y produce soluciones inclusive mejores que la ILS por su capacidad de iterar más veces en menos tiempo, por lo cual se recomienda la utilización de esta en casos donde se priorice la rapidez que la optimalidad.

7. Referencias

[1] M. Basset, A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem, *https://www.researchgate.net/profile/Mohamed-Abdel-Basset/publication/323864555_Hybrid_whale_optimization_algorithm_based_on_local_search_strategy_for_the_permutation_flow_shop_scheduling_problem/links/6032686845851589 39be8f4f/Hybrid-whale-optimization-algorithm-based-on-local-search-strategy-for-the-permutation-flow-shop-scheduling-problem.pdf*, 20-Mar-2018. [Online].

[2] S.M. Johnson, Optimal two and three stage production schedules with setup times included, Naval Res. Logist. 1 (1) (1954) 61–68.

[3] E. Taillard, Benchmarks for basic scheduling problems, European J. Oper. Res. 64 (2) (1993) 278–285.

[4] Andreas Fink. “Solving the Continuous Flow-Shop Scheduling Problem by Metaheuristics”. European Journal of Operational Research 151, pp 400–414, 2003.

[5] Lourenço, Helena & Martin, Olivier & Stützle, Thomas. (2003). Iterated Local Search. 10.1007/0-306-48056-5_11..

[6] Glover, Fred. (1990). Tabu Search: A Tutorial. Interfaces. 20. 74-94. 10.1287/inte.20.4.74.

[7] KHURSHID, BILAL(2021). An Improved Evolution Strategy Hybridization. With Simulated Annealing for Permutation Flow Shop Scheduling Problems.