

# DISEÑO Y ANÁLISIS DE ALGORITMOS

Algoritmos constructivos y  
búsquedas por entornos

*Parallel Machine Scheduling Problem  
with Dependent Setup Times*

Belén Melián Batista  
Curso 2023-2024

---

**OBJETIVO:**

Proponer, implementar y evaluar algoritmos constructivos y búsquedas por entornos para el *Parallel Machine Scheduling Problem with Dependent Setup Times*.

**TAREAS:**

Además de las tareas descritas en el presente documento, los estudiantes tendrán que realizar las modificaciones que se planteen durante la defensa final de la práctica.

**ENTREGAS PARCIALES Y MODIFICACIÓN FINAL:**

- Entrega parcial 1 (2 de abril): Voraz y fase constructiva de GRASP
- Entrega parcial 2 (9 de abril): GRASP con búsquedas locales
- Entrega final y defensa (16 de abril): Práctica completa (incluye GVNS) e informe en Latex (Overleaf) (Plantilla ejemplo).

**EVALUACIONES PARCIALES:**

Código fuente y defensa del trabajo realizado: hasta 10 puntos; si el día de la corrección falta algún código o este es incorrecto, la práctica se calificará como No Apta.

**EVALUACIÓN FINAL:**

Código fuente y memoria: hasta 4 puntos; si el día de la corrección falta algún código o este es incorrecto, la práctica se calificará como No apta.

Modificación propuesta el día de la corrección: hasta 4 puntos.

Defensa de la práctica: hasta 2 puntos.

**LENGUAJE DE PROGRAMACIÓN:**

Java o C++.

**PONDERACIÓN DE NOTAS DE LAS TRES ENTREGAS:**

Las defensas parciales supondrán un 25 % de la nota final de la práctica.

---

## Parallel machine scheduling problem with dependent setup times

En esta práctica estudiaremos un problema de secuenciación de tareas en máquinas paralelas con tiempos de setup dependientes de la secuencia. El objetivo del problema es asignar tareas a las máquinas y determinar el orden en el que deben ser procesadas de tal manera que la suma de los tiempos de finalización de todos los trabajos, es decir, el tiempo total de finalización (*TCT*), sea minimizado.

El tiempo de setup es el tiempo necesario para preparar los recursos necesarios (personas, máquinas) para realizar una tarea (operación, trabajo). En algunas situaciones,

los tiempos de setup varían según la secuencia de trabajos realizados en una máquina; por ejemplo en las industrias química, farmacéutica y de procesamiento de metales, donde se deben realizar tareas de limpieza o reparación para preparar el equipo para realizar la siguiente tarea.

Existen varios criterios de desempeño para medir la calidad de una secuenciación de tareas dada. Los criterios más utilizados son la minimización del tiempo máximo de finalización (*makespan*) y la minimización del TCT. En particular, la minimización del TCT es un criterio que contribuye a la maximización del flujo de producción, la minimización de los inventarios en proceso y el uso equilibrado de los recursos.

El problema abordado en esta práctica tiene las siguientes características:

- Se dispone de  $m$  máquinas paralelas idénticas que están continuamente disponibles.
- Hay  $n$  tareas independientes que se programarán en las máquinas. Todas las tareas están disponibles en el momento cero.
- Cada máquina puede procesar una tarea a la vez sin preferencia y deben usarse todas las máquinas.
- Cualquier máquina puede procesar cualquiera de las tareas.
- Cada tarea tiene un tiempo de procesamiento asociado  $p_j$ .
- Hay tiempos de setup de la máquina  $s_{ij}$  para procesar la tarea  $j$  justo después de la tarea  $i$ , con  $s_{ij} \neq s_{ji}$ , en general. Hay un tiempo de setup  $s_{0j}$  para procesar la primera tarea en cada máquina.
- El objetivo es minimizar la suma de los tiempos de finalización de los trabajos, es decir, minimizar el TCT.

El problema consiste en asignar las  $n$  tareas a las  $m$  máquinas y determinar el orden en el que deben ser procesadas de tal manera que se minimice el TCT.

El problema se puede definir en un grafo completo  $G = (V, A)$ , donde  $V = \{0, 1, 2, \dots, n\}$  es el conjunto de nodos y  $A$  es el conjunto de arcos. El nodo 0 representa el estado inicial de las máquinas (trabajo ficticio) y los nodos del conjunto  $I = \{1, 2, \dots, n\}$  corresponden a las tareas. Para cada par de nodos  $i, j \in V$ , hay dos arcos  $(i, j), (j, i) \in A$  que tienen asociados los tiempos de setup  $s_{ij}, s_{ji}$  según la dirección del arco. Cada nodo  $j \in V$  tiene asociado un tiempo de procesamiento  $p_j$  con  $p_0 = 0$ . Usando los tiempos de setup  $s_{ij}$  y los tiempos de procesamiento  $p_j$ , asociamos a cada arco  $(i, j) \in A$  un valor  $t_{ij} = s_{ij} + p_j, (i \in V, j \in I)$ .

Sea  $P_r = \{0, [1_r], [2_r], \dots, [k_r]\}$  una secuencia de  $k_r + 1$  tareas en la máquina  $r$  con el trabajo ficticio 0 en la posición cero de  $P_r$ , donde  $[i_r]$  significa el nodo (tarea) en la posición  $i_r$  en la secuencia  $r$ . Luego, el tiempo de finalización  $C_{[i_r]}$  del trabajo en la posición  $i_r$  se calcula como  $C_{[i_r]} = \sum_{j=1}^{i_r} t_{[j-1][j]}$ . Tenga en cuenta que en el grafo  $G$  representa la longitud de la ruta desde el nodo 0 al nodo  $[i_r]$ .

Sumando los tiempos de finalización de los trabajos en  $P_r$  obtenemos la suma de las longitudes de las rutas desde el nodo 0 a cada nodo en  $P_r$  ( $TCT(P_r)$ ) como:

$$TCT(P_r) = \sum_{i=1}^k C_{[i]} = kt_{[0][1]} + (k-1)t_{[1][2]} + \dots + 2t_{[k-2][k-1]} + t_{[k-1][k]} \quad (1)$$

Usando lo anterior, el problema se puede formular como encontrar  $m$  rutas simples disjuntas en  $G$  que comienzan en el nodo raíz 0, que juntas cubren todos los nodos en  $I$  y minimizan la función objetivo.

$$z = \sum_{r=1}^m TCT(P_r) = \sum_{r=1}^m \sum_{i=1}^{k_r} (k_r - i + 1)t_{[i-1][i]} \quad (2)$$

Tenga en cuenta que el coeficiente  $(k_r - i + 1)$  indica el número de nodos después del nodo en la posición  $i_r - 1$ .

### 0.0.1. Representación de las soluciones

Podemos generar un array por cada una de las máquinas,  $S = \{A_1, A_2, \dots, A_m\}$ . En ellos se insertarán las tareas a ser procesadas en cada máquina en el orden establecido.

## Un constructivo voraz

Un algoritmo constructivo voraz muy sencillo para este problema parte del subconjunto,  $S$ , formado por las  $m$  tareas con menores valores de  $t_{0j}$  asignadas a los  $m$  arrays que representan la secuenciación de tareas en las máquinas. A continuación, añade a este subconjunto, iterativamente, la tarea-máquina-posición que menor incremento produce en la función objetivo. El pseudocódigo de este algoritmo se muestra a continuación.

---

#### Algoritmo constructivo voraz

---

- 1: Seleccionar la  $m$  tareas  $j_1, j_2, \dots, j_m$  con menores valores de  $t_{0j}$  para ser introducidas en las primeras posiciones de los arrays que forman la solución  $S$ ;
  - 2:  $S = \{A_1 = \{j_1\}, A_2 = \{j_2\}, \dots, A_m = \{j_m\}\}$ ;
  - 3: **repeat**
  - 4:    $S^* = S$ ;
  - 5:   Obtener la tarea-máquina-posición que minimiza el incremento del  $TCT$ ;
  - 6:   Insertarla en la posición que corresponda y actualizar  $S^*$ ;
  - 7: **until** (todas las tareas han sido asignadas a alguna máquina)
  - 9: Devolver  $S^*$ ;
- 

Figura 1: Algoritmo constructivo voraz

# Implementación

Las instancias del problema se suministrarán en un fichero de texto con el siguiente formato:

$n$ : 40 //número de tareas  
 $m$ : 2 //número de máquinas  
 $P_i$ : 34 66 27 4 23 85 43 4 8 2 13 48 5 52 67 56 67 6 29 9 20 76 98 7 90 56 33 74 46 89 2 10 80 20 67 31 82 77 18 96 //tiempo de procesamiento de cada tarea  
 $S_{ij}$ : //matriz de dimensión  $(n + 1) \times (n + 1)$  con los tiempos de setup de una tarea a la siguiente, incluyendo la tarea 0 en la primera posición

## Tareas

- Diseñar e implementar el algoritmo constructivo voraz descrito en la figura 1.
- Diseñar e implementar un algoritmo GRASP para el problema. Implementar sólo la fase constructiva. La fase de mejora se definirá e implementará en el punto c).
- Diseñar e implementar un Método Multiarranque para el problema. Para ello se deberán definir las estructuras de entorno correspondientes a los siguientes 4 movimientos: intercambio de tareas entre máquinas y en la misma máquina y re-inserción de una tarea en otra posición de otra máquina o de la misma máquina.
- Diseñar e implementar una Búsqueda por Entorno Variable General para el problema.

## Qué debe presentar el alumno

- Código fuente, debidamente comentado, y fichero ejecutable.
- Una memoria en Latex en la que se describan brevemente los algoritmos diseñados enumerando las estructuras de datos usadas, las estructuras de entorno empleadas en las correspondientes búsquedas y cualquier elemento necesario para comprender el diseño propuesto. **Nota.-** Al aula virtual se debe subir un fichero pdf, en el que aparezca claramente el enlace de Overleaf a la memoria en Latex.
- Tablas o gráficas de resultados que muestren el comportamiento de los algoritmos sobre diferentes instancias del problema. A continuación se muestra un modelo de tablas de resultados que el alumno puede usar. No obstante, se trata solo de un modelo que puede modificarse si se cree que otro es mejor.

Algoritmo voraz				
Problema	$n$	Ejecución	$TCT$	$CPU$
$ID$		1		
$ID$		2		
$ID$		3		
$ID$		4		
$ID$		5		
...	...	...	...	...

Cuadro 1: Algoritmo voraz. Tabla de resultados

Multiarranque				
Problema	$n$	Ejecución	$TCT$	$CPU$
$ID$		1		
$ID$		2		
$ID$		3		
$ID$		4		
$ID$		5		
...	...	...	...	...

Cuadro 2: Multiarranque. Tabla de resultados

GRASP					
Problema	$n$	$ LRC $	Ejecución	$TCT$	$CPU$
$ID$		2	1		
$ID$		2	2		
$ID$		2	3		
$ID$		2	4		
$ID$		2	5		
$ID$		3	1		
$ID$		3	2		
$ID$		3	3		
$ID$		3	4		
$ID$		3	5		
...	...	...	...	...	...

Cuadro 3: GRASP. Tabla de resultados

VNS					
Problema	$m$	$k_{max}$	Ejecución	$TCT$	$CPU$
<i>ID</i>		2	1		
<i>ID</i>		2	2		
<i>ID</i>		2	3		
<i>ID</i>		2	4		
<i>ID</i>		2	5		
<i>ID</i>		3	1		
<i>ID</i>		3	2		
<i>ID</i>		3	3		
<i>ID</i>		3	4		
<i>ID</i>		3	5		
...	...	...	...	...	...

Cuadro 4: VNS. Tabla de resultados