Problema de secuenciación de tareas en máquinas paralelas con objetivo de latencia

Carlos García González, Francisco Javier Mendoza Álvarez, Aduanich Rodríguez Rodríguez, Alejandro David Carrillo Padrón, David de León Rodríguez[a]

[a] *Universidad de La Laguna*
Departamento de Ingeniería Informática y de Sistemas

1. Introducción

La gran diversidad de problemas reales del ámbito científico, de producción de bienes y de distribución de recursos que surgen en la mayoría de las empresas han motivado un gran desarrollo de técnicas y metodologías de solución para problemas como el que abordamos en este informe. El problema de secuenciación de tareas con objetivo de latencia es un contratiempo común en múltiples ámbitos, no solo en el campo de la informática, es por esto que es interesante abordar este problema. Ante todo se debe plantear el concepto de secuenciación, el cual es un proceso para la toma de decisiones que se centra en la asignación de recursos a tareas en periodos temporales determinados y su meta es optimizar esa colocación en base a uno o varios objetivos dados.

2. Descripción el Problema

El problema de secuenciación de tareas en máquinas con objetivo de latencia se centra en tratar de encontrar una distribución de las tareas a ejecutar entre las máquinas disponibles de forma que la suma de latencias de las tareas sea mínimo.

Se dispone de m máquinas paralelas y una lista con n tareas. Se deben asignar las n tareas entre las máquinas de manera que la suma de latencias sea menor, ejecutándose cada tarea una única vez en alguna de las m máquinas.

Además, entre la ejecución de cada tarea hay un tiempo de preparación previo que varía en función de la tarea ejecutada y la que será ejecutada a continuación. Por lo tanto, el orden en que se ejecutan las tareas influye en el tiempo de latencia total de cada máquina, puesto que el tiempo de espera entre la tarea i y la tarea j es distinto del tiempo de espera entre j e i.

A la hora de abordar este problema también es necesario concretar qué se entiende por latencia. Llamamos latencia al tiempo que espera cada tarea a ser ejecutada más el tiempo que tarda en ejecutarse. Por tanto, el tiempo de latencia total de una máquina correspondería con la suma de los tiempos de latencia de todas las tareas asignadas a esa máquina en el orden de ejecución correspondiente más los tiempos de espera concretos entre las ejecuciones de cada una de las tareas.

3. Representación y codificación de soluciones. Estructuras de entorno y evaluación de la función objetivo

En cuanto a la representación de la solución elegida barajamos varias soluciones pero en última estancia escogimos utilizar un *vector* de máquinas. A su vez, las máquinas tendrían almacenadas otro *vector* que contendría las tareas a asignadas a dicha máquina por orden de ejecución.

Escogimos esta manera de representar la solución dado que nos permite cambiar las estructuras de entorno de una forma fácil e intuitiva, además de que se mejora el uso de memoria.

En cuanto a la función objetivo debemos concretar puesto que es la que usaremos para determinar el tiempo de latencia total. La función objetivo se aplicara a todas las listas de ejecución de las máquinas para calcular la latencia total de cada una. De entre todas esas seleccionaremos el valor máximo y nos enfocaremos en buscar soluciones que minimicen ese valor máximo encontrado. El problema queda definido así como un problema cuyo objetivo es encontrar el valor de latencia máximo y minimizarlo.

La función objetivo quedaría descrita de la siguiente manera:

$$LatenciaTotal = P_0 + \sum_{i=1}^{n}(S_{ij} + P_j)$$

Donde:

- n es el número de tareas.
- P_x es el tiempo de ejecución de la tarea x.
- S_{ij} es el tiempo de preparación entre la tarea i y la tarea j.

Una vez definida la función objetivo podemos empezar a definir las estructuras de entorno que utiizarán los distintos algortimos para tratar de minimizar el valor del máximo tiempo de latencia de las máquinas paralelas.

4. Definición de las estructuras de entorno y evaluación de los movimientos

Una estructura de entorno es una función que asocia a cada solución un conjunto de soluciones cercanas en algún sentido. Queda a elección del programador establecer que se entiende por cercanía. Generalmente, se definen por movimientos aplicados a una solución que generen soluciones diferentes.

En el desarrollo de nuestro programa definimos distintas estructuras de entorno para poder encontar más fácilmente diferentes soluciones que de otro modo no visitaríamos. Al utilizar distintas estructuras te permite visitar más vecinos diferentes para buscar la mejor solución posible.

Para la resuloción de este problema definimos e implementamos tres estructuras de entorno diferentes para asegurar que los algoritmos heurísticos que utilizaremos para calcular la solución nos proporcionarán una que sea lo más acercada a la óptima. Las estructuras de entorno se definen a continuación.

4.1. Intra-máquina

Consiste en intercambiar dos tareas dentro de la lista de ejecución de la misma máquina.

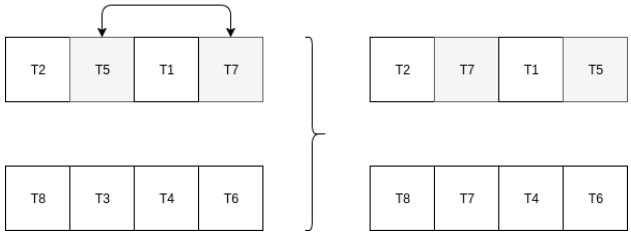

Figura 1: Gráfica de estructura de entorno intra-máquina

4.2. Entre-máquinas

Intercambia tareas entre máquinas, dejando las listas de ejecución de las máquinas involucradas con el mismo número de tareas.

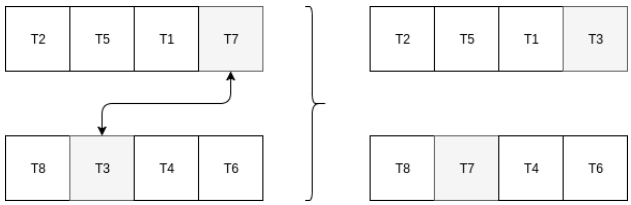

Figura 2: Gráfica de estructura de entorno entre-máquinas

4.3. Re-inserción

Se trata de sacar una tarea de la lista de ejecución de una máquina y meterla en otra, sin intercambio. Es decir, después del movimiento, una máquina quedaría con una tarea menos, y otra con una tarea más.

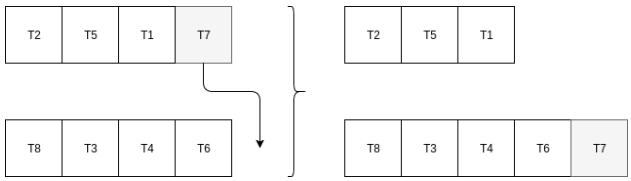

Figura 3: Gráfica de estructura de entorno reinserción

5. Experiencia Computacional

5.1. Objetivos de la experiencia computacional

5.2. Descripción de las instancias utilizadas

5.3. Análisis de algoritmos y comparativa.

Se deben usar test estadísticos nos paramétricos (Friedman and Wilkoxon) para determinar si hay diferencias significativas entre los algoritmos. Se puede hacer uso de r (https://www.rstudio.com/) o de cualquier otro software, tipo XLS Stat.

5.3.1. Algoritmo constructivo determinista

El algoritmo voraz consiste en ir añadiendo las tareas en las máquinas de forma que la latencia del sistema crezca lo menor posible. Esto se hace añadiendo primero las tareas que tengan un tiempo de ejecución menor, determinando en qué máquina deben introducirse y en qué orden según el valor de la función objetivo.

Con este algoritmo obtendremos una solución no óptima pues no se hace una mejora de la solución obtenida.

5.3.2. GRASP

El algoritmo GRASP consiste en iterar contruyendo soluciones greedy de forma aleatoria para luego mejorar la solución obtenida con una búsqueda local.

De esta forma tenemos dos etapas en cada iteración, la de construcción y la de mejora. En la primera se generan la solución greedy añadiendo tareas a las máquinas de una lista de las tareas ordenadas por su tiempo de ejecución. Las tareas se añaden según la calidad de la solución que se va formando. La variabilidad de la lista de tareas candidatas se obtiene introduciendo las tareas en una lista restringida de candidatos elegidos al azar al crear la solución.

5.3.3. Multiarranque

5.3.4. VNS

5.3.5. Búsqueda Tabú

5.3.6. LNS

· Link