Universidad de Valladolid

Algoritmos y Computación

# Job Shop Scheduling

*Sergio García Prado*

Seguimiento del trabajo en:
github.com/garciparedes/Job-Shop-Scheduling-NP-Complete

December 26, 2015

# Contents

# 1 Introducción

## 1.1 Definición del problema

Job Shop Scheduling es un problema de optimización que se estudia en Ciencias de la Computación e Investigación de Operaciones. Ahora describiremos el problema y el objetivo que se pretende conseguir:

Supongamos que tenemos $n$ trabajos a realizar, los cuales denotamos como $J_i$ tal que $1 \leq i \leq n$ cada uno de los cuales de distinta duración. También tenemos $r$ máquinas con las que realizar los trabajos que denotaremos por $M_j$ tal que $1 \leq j \leq r$. Cada una de estas es la encargada de realizar una operación concreta necesaria para finalizar cada uno de los trabajos. A cada una de estas operaciones las denotaremos por $O_j i$ que representa la operación que se debe realizar en la máquina $M_j$ del trabajo $J_i$.

En Job Shop Scheduling el orden de las operaciones necesarias para completar el i-ésimo trabajo importa y no se puede modificar.

El objetivo será tratar de minimizar al máximo posible el tiempo necesario para completar todas las operaciones. Para ello trataremos el problema como de decisión, es decir, buscaremos la respuesta la siguiente pregunta:
**¿Se pueden resolver los $i$ trabajos a partir de las $j$ máquinas en un tiempo menor o igual que $r$?**

## 1.2 Open Shop, Job Shop y Flow Shop

Estos problemas tres problemas son muy similares entre si, ya que todos están referidos a la planificación de trabajos en más de una máquina.

- **Open Shop**: El orden de realización de las operaciones **no** importa.
- **Job Shop**: El orden de realización de las operaciones importa.
- **Flow Shop**: Cada trabajo tiene exactamente una operación por cada máquina, y todos los trabajos siguen el mismo orden de operaciones.

Hoy en día Job Shop Scheduling es presentado como un problema online, lo que quiere decir que cuando le llega un nuevo trabajo, el algoritmo debe devolver una decisión antes de conocer el próximo trabajo. A pesar de que la formulación actual sea como un problema online para el análisis de reducción no se tendrá en cuenta este caso, es decir, supondremos que conocemos a priori todos los trabajos que planificar (problema offline).

# 2 Reducción

Para demostrar que el problema Job Shop Scheduling pertenece a la clase NP-Complete nos basaremos en unos problemas cuya pertenencia a dicha clase ya está demostrada, por lo cual nos bastará demostrar que JobShop se puede reducir a estos.

El método que utilizaremos será reducción polinómica, lo que quiere decir que tendremos que conseguir demostrar que existe un método de la clase P que permita transformar nuestro problema a un problema de la clase NP-Complete. Con esto se consigue que si se consiguiera demostrar que un problema de la clase NP-Complete pertenece también a la clase P, todos los problemas de la clase NP-Complete quedarían sistemáticamente demostrados como pertenecientes a la clase P por transitividad.

## 2.1 Partición

**Partición** es un problema NP-completo, que visto como un problema de decisión, consiste en decidir si, dado un multiconjunto (conjunto en el cual cada miembro del mismo tiene asociada una multiplicidad) de números enteros, puede éste ser particionado en dos "mitades" tal que sumando los elementos de cada una, ambas den como resultado la misma suma.

Más precisamente, dado un multiconjunto S de enteros: ¿existe alguna forma de particionar S en dos subconjuntos S1 y S2, tal que la suma de los elementos en S1 sea igual que la suma de los elementos en S2?

La solución con programación dinámica existente para resolver el problema de suma de subconjuntos, utilizando tiempo pseudo-polinómico, también es aplicable al problema de partición.

**Tiempo Pseudo-Polinómico**: un algoritmo numérico se ejecuta en tiempo pseudo-polinómico si su tiempo de ejecución es polinómico en el valor numérico de la entrada, pudiendo ser este valor exponencial en el largo de la entrada, es decir, en el número de dígitos que la conforman. Los problemas NP-completos con algoritmos que se ejecutan en tiempo pseudo-polinómico se denominan NP-completos débiles. Un problema NP-completo se denomina NP-completo fuerte si se puede demostrar que no puede ser resuelto por algoritmos pseudo-polinómicos, salvo que se cumpla que P=NP.

## 2.2 Job Shop

## 3 Bibliografía

-