UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

CALIDAD DE ÓPTIMOS LOCALES PARA PROBLEMAS DE PROGRAMACIÓN DE LA PRODUCCIÓN EN MÁQUINAS PARALELAS

TESIS PARA OPTAR AL GRADO DE DOCTOR EN SISTEMAS DE INGENIERÍA

FELIPE TOMÁS MUÑOZ VALDÉS

PROFESOR GUÍA:
JOSE CORREA HAEUSSLER

MIEMBROS DE LA COMISIÓN:
RODRIGO CARRASCO SCHMIDT
RAFAEL EPSTEIN NUMHAUSER
JOSÉ VERSCHAE TANNENBAUM

Este trabajo ha sido parcialmente financiado por Universidad del Bío-Bío; CONICYT, Programa de Formación de Capital Humano Avanzado; Núcleo Milenio Información y Coordinación en Redes

SANTIAGO DE CHILE
2016

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE DOCTOR EN SISTEMAS DE INGENIERÍA
POR: FELIPE TOMÁS MUÑOZ VALDÉS
FECHA: 2016
PROF. GUÍA: SR. JOSE CORREA HAEUSSLER

CALIDAD DE ÓPTIMOS LOCALES PARA PROBLEMAS DE PROGRAMACIÓN DE LA PRODUCCIÓN EN MÁQUINAS PARALELAS

En este trabajo se estudia la calidad que ofrecen las soluciones óptimas locales para problemas de programación de tareas en máquinas en paralelo. Los ambientes considerados son máquinas idénticas, idénticas restringidas, uniformes restringidas y no-relacionadas. El objetivo considerado es la minimización del tiempo ponderado de completación. Para estudiar la calidad de los óptimos locales se determinan los factores de aproximación para las soluciones localmente óptimas de los vecindarios de inserción (jump) e intercambio (swap).

Los resultados indican que para los ambientes de máquinas paralelas uniformes y no-relacionadas, el costo de cualquier óptimo local se encuentra alejado a lo más en un factor $2,618$ con respecto al costo del óptimo. Si solo se considera la minimización del tiempo de completación, se tiene que el factor es 2. El mismo resultado se obtuvo para el ambiente de máquinas uniformes con tareas unitarias, para los casos ponderado y no ponderado.

Por otra parte, para el problema de máquinas paralelas idénticas restringidas, se determinó que el factor de aproximación se encuentra entre $1,75$ y $1,809$. Para el caso no ponderado este factor se encuentra entre $1,5333$ y $1,618$. Para el caso de tareas unitarias, donde el objetivo es la minimización del tiempo ponderado de completación, se determinó que el factor de aproximación se encuentra entre $1,5333$ y $1,618$. Mientras que para el caso no ponderado se tienen evidencias que indican que el factor de aproximación es $1,5333$.

Dedicado a mi esposa e hijas.

Agradecimientos

A José R. Correa, muchas gracias por todo el tiempo, apoyo y consejos. A José Verschae y Victor Verdugo por los valiosos consejos.

Finalmente, agradezco el financiamiento otorgado por Universidad del Bío-Bío; CONICYT, Programa de Formación de Capital Humano Avanzado; Núcleo Milenio Información y Coordinación en Redes: ICM/FIC RC130003.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

En este capítulo se describen los principales conceptos para el desarrollo de esta tesis. Se describe el problema en estudio, sus propiedades, variantes y complejidad. Se describe la búsqueda local y sus características principales. Se revisan algoritmos de aproximación propuestos, con énfasis en resultados relacionados a búsqueda local.

1.1. Scheduling

Los problemas de programación de tareas en máquinas (machine scheduling), tratan sobre la asignación de recursos escasos a través del tiempo. Un ejemplo clásico, son las líneas de producción de una fábrica que debe procesar productos para sus clientes.

En general, una instancia de un problema de programación de tareas consta de un conjunto de tareas $\mathcal{J}$ y un conjunto de máquinas $\mathcal{M}$. Una solución del problema, es un programa o schedule, el cual especifica en qué máquina, y en qué instante de tiempo se procesa cada tarea. Esta descripción deja implícita la necesidad de secuenciar las tareas, es decir, establecer el orden en que las tareas son procesadas por las máquinas. Baker y Trietsch [9], describen una solución por medio de dos preguntas: ¿qué recurso debería ser asignado para procesar cada tarea?, ¿cuándo debería ser procesada cada tarea?.

Existe mucha literatura con respecto a programación de tareas en máquinas. Partiendo de los textos clásicos de Conway, Maxwell y Miller [22]; Baker [8]. A los textos más recientes Brucker [11]; Pinedo [58]; Baker y Trietsch [9].

Existen diferentes formas de clasificar los problemas de programación de tareas, una de ellas es considerar la cantidad de operaciones que requiere cada tarea. Según esta clasificación, existen dos tipos de problemas:

- **Scheduling de una operación**. Donde la operación debe ser realizada en una máquina o en máquinas en paralelo.
- **Scheduling de múltiples operaciones**. Donde las operaciones deben ser procesadas

en un taller, que cuenta con varias estaciones de trabajo, cada estación de trabajo realiza una operación sobre la tarea. Existen tres tipos de talleres:

- Taller de flujo (flow shop), las tareas tienen la misma secuencia con respecto a las estaciones de trabajo.
- Taller de trabajos (job shop), las tareas tienen distinta secuencia con respecto a las estaciones de trabajo.
- Taller abierto (open shop), el orden en el cual se realizan las operaciones sobre las tareas es arbitrario.

Otra forma de clasificar los problemas de scheduling, es con respecto al dinamismo que presenta la llegada de tareas. Según esta clasificación, existen dos tipos de problemas:

- **Scheduling offline**. Toda la información o data requerida para resolver una instancia, es conocida a priori.
- **Scheduling online**. Las tareas llegan o arriban al sistema una a una. La solución se construye, sin saber lo que pasará con respecto a la llegada de nuevas tareas. Este tipo de problemas puede ser visto como un scheduling con información incompleta. Donde las decisiones tienen que hacerse sin conocer la instancia completa.

Otra forma de clasificar los problemas, es con respecto al tipo de variables con que se modelan los parámetros del problema. Según esta clasificación, existen dos tipos de problemas:

- **Scheduling determinístico**. Todos los parámetros del sistema, y la información sobre las tareas son conocidos a priori y modelados por variables deterministas.
- **Scheduling estocástico**. Al menos un parámetro del sistema, o alguna información sobre las tareas se modelan como variables aleatorias.

Estos problemas también pueden ser clasificados con respecto a la cantidad de tomadores de decisiones que se considera. Según esta clasificación, existen dos tipos de problemas:

- **Scheduling centralizado**. Toda la información requerida para resolver el problema es recogida por un tomador de decisiones, cuyo objetivo es optimizar algún criterio.
- **Scheduling descentralizado**. La información requerida para resolver el problema se encuentra distribuida entre las distintas partes o componentes que forman el problema. Estas partes tienen autonomía en la toma de decisiones, optimizando su beneficio.

Una forma de scheduling descentralizado, son los juegos de scheduling (basados en teoría de juegos). En este tipo de problemas, para el caso de máquinas paralelas, las tareas son administradas por agentes que toman la decisión de dónde asignar la tarea, optimizando su propio beneficio, sin considerar el objetivo global del sistema (costo social). Desde el punto de vista de decisión centralizado, esta forma de abordar un problema llevará a una ineficiencia. Esta ineficiencia se mide con el precio de la anarquía, que es el cociente entre el peor equilibrio de Nash (peor solución del juego) y el óptimo global (social), obtenido a partir de la versión centralizada del problema.

De acuerdo a las clasificaciones presentadas anteriormente, los problemas estudiados en esta tesis corresponden a problemas de **scheduling de una operación, offline, determi-**

nístico y centralizado.

1.2. Descripción del problema

A continuación se describe el problema de scheduling que se estudia en testa tesis. Dada la gran diversidad de problemas que se generan al variar cada una de las partes involucradas en el problema, en el año 1979, Graham et al. [41] introdujeron la notación de tres campos, representada por $\alpha|\beta|\gamma$. En el campo α se indica el ambiente de las máquinas, en el campo β se indican las características de las tareas, y en γ el (los) objetivo(s) que se desea optimizar.

Sea $\mathcal{J} = \{1, \ldots, n\}$ el conjunto de tareas y $\mathcal{M} = \{1, \ldots, m\}$ el conjunto de máquinas. Cada tarea $j \in \mathcal{J}$ requiere un tiempo de proceso no negativo p_{ij} en la máquina $i \in \mathcal{M}$. Las tareas deben ser programadas sin interrupción en una máquina, y cada máquina puede procesar una tarea a la vez. Además todas las tareas y máquinas se encuentran disponibles desde el inicio. De acuerdo a las características de las máquinas, se tienen los siguientes ambientes de trabajo:

- **Máquinas paralelas idénticas** (P): cada tarea tiene el mismo tiempo de proceso en todas las máquinas; los tiempos de proceso son $p_{ij} = p_j, \forall i \in \mathcal{M}$.
- **Máquinas paralelas idénticas restringidas** (RP): este ambiente es una extensión del ambiente P. Adicionalmente se considera que la tarea j puede ser procesada por un conjunto restringido de máquinas $\mathcal{M}_j \subseteq \mathcal{M}$; los tiempos de proceso son $p_{ij} = p_j, \forall i \in \mathcal{M}_j$; $p_{ij} = \infty, \forall i \notin \mathcal{M}_j$.
- **Máquinas paralelas uniformes o relacionadas** (Q): las máquinas tienen una velocidad $s_1, \ldots, s_m$, y cada tarea tiene un requerimiento de proceso p_j; los tiempos de proceso son $p_{ij} = \frac{p_j}{s_i}$.
- **Máquinas paralelas relacionadas restringidas** (RQ): este ambiente es una extensión del ambiente Q. Adicionalmente se considera que la tarea j puede ser procesada por un conjunto restringido de máquinas $\mathcal{M}_j \subseteq \mathcal{M}$; los tiempos de proceso son $p_{ij} = \frac{p_j}{s_i}, \forall i \in \mathcal{M}_j$; $p_{ij} = \infty, \forall i \notin \mathcal{M}_j$.
- **Máquinas paralelas no-relacionadas** (R): el tiempo de procesamiento de la tarea j en la máquina i es arbitrario (depende de la tarea y la máquina). Este es el caso más general en ambientes de máquinas en paralelo. Se puede considerar que todos los ambientes de máquinas paralelas, son un caso particular de máquinas paralelas no-relacionadas.

De acuerdo a la notación de scheduling introducida por Graham et al. [41], para el caso de máquinas en paralelo, cuando el número de máquinas es parte de la entrada se representa por P, Q y R, si el ambiente considera máquinas idénticas, relacionadas o no-relacionadas respectivamente. Cuando la cantidad de máquinas es fija (m) se usa la notación P_m, Q_m y R_m respectivamente. De manera similar se usa el subíndice m para los casos restringidos.

Denotamos por C_j y w_j al tiempo de completación, y peso o importancia relativa de la tarea j. Existen tres objetivos directamente relacionados al tiempo de completación:

- **Tiempo total de completación** ($\sum_{j\in\mathcal{J}} C_j$). El objetivo es minimizar la suma de los tiempos en que cada tarea es completada.
- **Tiempo ponderado de completación** ($\sum_{j\in\mathcal{J}} w_j C_j$). El objetivo es minimizar la suma ponderada de los tiempos en que cada tarea es completada.
- **Makespan** ($C_{\text{máx}} = \text{máx}_{j\in\mathcal{J}}\{C_j\}$). El objetivo es minimizar el momento en el tiempo en que la última tarea es completada.

En la literatura (por ejemplo Leung y Li [55], Pinedo [58]) se representan los ambientes restringidos desde la perspectiva de las tareas usando la notación $P|\mathcal{M}_j|\sum w_j C_j$, donde $\mathcal{M}_j$ en el segundo campo representa la elegibilidad de máquina de la tarea j. Sin embargo, respetando la notación propuesta por Graham et al. [41], y la definición que realiza sobre el campo β, es más adecuado usar $RP||\sum w_j C_j$. Ya que son las máquinas las que presentan restricciones para procesar las tareas.

Leung y Li [55] presentan una revisión de algoritmos de aproximación para distintos problemas en ambientes de máquinas paralelas restringidas, donde el objetivo es minimizar el makespan. Incluyen casos de online y offline scheduling, con tareas interrumpibles y no interrumpibles. Plantean que los casos especiales de ambientes restringidos más estudiados son:

- Conjuntos restringidos anidados. Para cada par $\mathcal{M}_j$ y $\mathcal{M}_k$, se tiene que: $\mathcal{M}_j$ y $\mathcal{M}_k$ son disjuntos; o $\mathcal{M}_j \subseteq \mathcal{M}_k$; o $\mathcal{M}_k \subseteq \mathcal{M}_j$.
- Conjuntos restringidos inclusivos. Para cada par $\mathcal{M}_j$ y $\mathcal{M}_k$, se tiene que: $\mathcal{M}_j \subseteq \mathcal{M}_k$; o $\mathcal{M}_k \subseteq \mathcal{M}_j$.

Una instancia $\mathcal{I}$ del problema $P||\sum w_j C_j$, queda completamente definida por $\mathcal{J}$, $\mathcal{M}$, $\boldsymbol{w}$ y $\boldsymbol{p}$. Donde $\boldsymbol{w}$ y $\boldsymbol{p}$ representan el vector de pesos y tiempos de proceso de las tareas respectivamente. Para los problemas $RP||\sum w_j C_j$ y $RQ||\sum w_j C_j$, es necesario incluir los conjuntos $\mathcal{M}_j$, $\forall j \in \mathcal{J}$. Además para el problema $RQ||\sum w_j C_j$ es necesario definir el vector de velocidad de las máquinas $\boldsymbol{s}$. Para el problema $R||\sum w_j C_j$ se debe considerar que los tiempos de proceso se describen por una matriz $\boldsymbol{P}$. Otras variantes del problema se pueden construir al considerar que los tiempos de proceso y/o los pesos de las tareas son constantes.

Como se verá en las siguientes secciones, la calidad de los óptimos locales para el objetivo de minimizar el makespan en ambientes de máquinas paralelas, ha sido ampliamente estudiado. Por lo que el desarrollo de la tesis se enfocó en los objetivos: minimizar el tiempo ponderado de completación $\sum w_j C_j$, y minimización del tiempo total de completación $\sum C_j$. Los problemas estudiados en esta tesis, y sus variantes se presentan en la tabla 1.3.

1.2.1. Secuenciamiento

En esta sección se presentan las reglas de secuenciamiento para las tareas. Estas reglas permiten construir una secuencia óptima, dada una determinada asignación de tareas a las máquinas.

Smith [67] presenta resultados sobre el secuenciamiento de tareas en problemas de pro-

gramación en una máquina. A continuación se presenta una adaptación de los teoremas planteados por Smith. Para ello, se define:

$\boldsymbol{z}$: una solución factible del problema;

$N_i(\boldsymbol{z})$: conjunto de tareas asignadas a la máquina i en la solución $\boldsymbol{z}$.

Teorema 1.1 *Regla WSPT (Adaptado de Smith [67])*[1] .
El mínimo tiempo ponderado de completación en la máquina i, $\sum_{j\in N_i(\boldsymbol{z})} w_j C_j$. *Se obtiene secuenciando las tareas en orden decreciente según los cocientes* $\frac{w_j}{p_{ij}}$.

Corolario 1.2 *Regla SPT (Adaptado de Smith [67]).*
El mínimo tiempo de completación en la máquina i, $\sum_{j\in N_i(\boldsymbol{z})} C_j$. *Se obtiene secuenciando las tareas en orden creciente según los tiempos de proceso* p_{ij}.

Estas reglas de secuenciamiento, pueden ser utilizadas para definir relaciones de precedencia entre las tareas. De tal manera que el símbolo $k \prec_i j$ indica que la tarea k precede a la tarea j, si ambas están asignadas a la máquina i. Por otra parte $k \succ_i j$ indica que la tarea k sucede a la tarea j, si ambas están asignadas a la máquina i. Los símbolos $\preceq_i$ y $\succeq_i$, además incluyen la tarea j. Al aplicar las reglas WSPT o SPT, si existe empate entre dos o más tareas, su precedencia se define en forma arbitraria, sin afectar el costo.

1.2.2. Función objetivo

Considerando el secuenciamiento definido por la regla WSPT (Teorema 1.1), se tiene que el costo de cualquier solución $\boldsymbol{z}$ de una instancia $\mathcal{I}$ del problema $R||\sum w_j C_j$ puede ser determinado por:

$$\begin{aligned} C(\boldsymbol{z}) &= \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\preceq_i j}} w_j p_{ik} \\ &= \eta(\boldsymbol{z}) + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\prec_i j}} w_j p_{ik} \end{aligned} \tag{1.1}$$

o bien,

$$\begin{aligned} C(\boldsymbol{z}) &= \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succeq_i j}} w_k p_{ij} \\ &= \eta(\boldsymbol{z}) + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succ_i j}} w_k p_{ij} \end{aligned} \tag{1.2}$$

donde, $\eta(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} w_j p_{ij}$

[1] Al cociente $\frac{w_j}{p_{ij}}$ se le denomina cociente de Smith (CS) o Smith's ratio. A la regla WSPT también se le denomina regla de Smith.

1.2.3. Invariantes

A continuación se presenta una propiedad de los problemas en máquinas paralelas. La que explica la equivalencia entre los problemas que se presentan en la tabla 1.1.

Lema 1.3 *Cualquier problema de máquinas paralelas, donde el objetivo sea* $\sum w_j C_j$*, es invariante frente a un re-escalamiento en los tiempos de proceso y/o pesos de todas las tareas.*

Demostración. Sea $\mathcal{I}$ una instancia del problema $R||\sum w_j C_j$; $\mathcal{I}'$ una instancia obtenida desde $\mathcal{I}$ al multiplicar todos los tiempos de proceso por alguna constante positiva c_1, y los pesos de todas las tareas por otra constante positiva c_2. Esto es $p'_{ij} = c_1 p_{ij}$ y $w'_j = c_2 w_j$ para todo para todo $i \in \mathcal{M},\ j \in \mathcal{J}$.
Sea $\boldsymbol{z}$ alguna solución de $\mathcal{I}$ con costo

$$C(\mathcal{I}(\boldsymbol{z})) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\preceq j}} w_j p_{ik}$$

Por otra parte, el costo de la solución $\boldsymbol{z}$, en la instancia $\mathcal{I}'$ es:

$$C(\mathcal{I}'(\boldsymbol{z})) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\preceq j}} w'_j p'_{ik} = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\preceq j}} c_1 c_2 w_j p_{ik} = c_1 c_2 C(\mathcal{I}(\boldsymbol{z}))$$

Luego, para cualquier solución $\boldsymbol{z}$ de una instancia $\mathcal{I}$ del problema $R||\sum w_j C_j$, se tiene que $C(\mathcal{I}'(\boldsymbol{z})) = c_1 c_2 C(\mathcal{I}(\boldsymbol{z}))$.
Este resultado se puede generalizar para cualquier ambiente de máquinas paralelas. Porque los ambientes de máquinas idénticas (P), idénticas restringidas (RP), uniformes (Q) y uniformes restringidas (RQ) son casos particulares del ambiente de máquinas paralelas no-relacionadas (R). □

Corolario 1.4 *Sea* $\mathcal{P}$ *cualquier problema de máquinas paralelas, donde el objetivo es* $\sum w_j C_j$*,* $\boldsymbol{x}$ *algún óptimo local,* $\boldsymbol{x}^*$ *algún óptimo global. El factor de aproximación* $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)}$ *no se altera frente a un re-escalamiento en los tiempos de proceso y/o pesos de todas las tareas.*

1.3. Complejidad

Desde el punto de vista de la teoría de complejidad computacional, los problemas de máquinas paralelas que consideran el objetivo de minimizar $\sum w_j C_j$ son NP-hard, es decir, a menos que P=NP, no existen algoritmos eficientes para encontrar la solución óptima.

En 1979, Graham et al. [41] publicó la jerarquía de complejidad para problemas de scheduling. Donde se presentan las reducciones elementales entre los problemas de programación de tareas. La idea básica detrás de esta jerarquía, se fundamenta en los conceptos de complejidad y reducibilidad. Con respecto a los ambientes de máquinas estudiados, y suponiendo que la función objetivo a optimizar es la misma, se tiene que los problemas de máquinas

Problema	Equivalente
$P\|w_j = w\|\sum w_j C_j$	$P\|\|\sum C_j$
$P\|p_j = p\|\sum w_j C_j$	$P\|p_j = 1\|\sum w_j C_j$
$P\|p_j = p, w_j = w\|\sum w_j C_j$	$P\|p_j = 1\|\sum C_j$
$RP\|w_j = w\|\sum w_j C_j$	$RP\|\|\sum C_j$
$RP\|p_j = p\|\sum w_j C_j$	$RP\|p_j = 1\|\sum w_j C_j$
$RP\|p_j = p, w_j = w\|\sum w_j C_j$	$RP\|p_j = 1\|\sum C_j$
$Q\|w_j = w\|\sum w_j C_j$	$Q\|\|\sum C_j$
$Q\|p_j = p\|\sum w_j C_j$	$Q\|p_j = 1\|\sum w_j C_j$
$Q\|p_j = p, w_j = w\|\sum w_j C_j$	$Q\|p_j = 1\|\sum C_j$
$RQ\|w_j = w\|\sum w_j C_j$	$RQ\|\|\sum C_j$
$RQ\|p_j = p\|\sum w_j C_j$	$RQ\|p_j = 1\|\sum w_j C_j$
$RQ\|p_j = p, w_j = w\|\sum w_j C_j$	$RQ\|p_j = 1\|\sum C_j$
$R\|w_j = w\|\sum w_j C_j$	$R\|\|\sum C_j$
$R\|p_j = p\|\sum w_j C_j$	$R\|p_j = 1\|\sum w_j C_j$
$R\|p_j = p, w_j = w\|\sum w_j C_j$	$R\|p_j = 1\|\sum C_j$

Tabla 1.1: Equivalencia entre problemas.

paralelas donde m (cantidad de máquinas) es parte de la entrada (P, Q o R), son igual o más complejos que si se considera m fijo (P_m, Q_m o R_m). También, se puede establecer que los problemas en máquinas paralelas no-relacionadas (R) son igual o más complejos que en el ambiente Q, y estos a su vez son igual o más complejos que en el ambiente P. Con respecto a las funciones objetivo, y suponiendo el mismo ambiente de máquinas, $\sum w_j C_j$ es una función objetivo que induce mayor complejidad que $\sum C_j$. Para el makespan, no es posible establecer una relación directa con el objetivo $\sum w_j C_j$, pero se sabe que induce más complejidad que el objetivo $\sum C_j$.

Los problemas de máquinas paralelas que consideran como objetivo la minimización del tiempo de completación total $\sum C_j$, se pueden resolver en tiempo polinomial, independiente del ambiente de las máquinas. El problema $P||\sum C_j$ se puede resolver en $O(n \log n)$ usando la regla SPT[2] (Conway et al. [22]). El problema $Q||\sum C_j$ se puede resolver en $O(n \log nm)$ usando una generalización de la regla SPT, donde las tareas se asignan secuencialmente a la máquina que permite un menor tiempo de completación, de acuerdo a su carga de trabajo y velocidad (Conway et al. [22]; Horowitz y Sahni [46]). Para el problema $R||\sum C_j$, Horn [45], plantea que el problema puede ser resuelto por un emparejamiento bipartito en $O(\text{máx}\{n^3, mn^2\})$.

Para los problemas de máquinas paralelas, si el objetivo es minimizar la suma ponderada de tiempos de completación ($\sum w_j C_j$) o el makespan ($C_{\text{máx}}$), el problema es NP-hard. Inclu-

[2]Shortest processing time first.

so cuando se tienen 2 máquinas idénticas (Bruno, Coffman Jr. y Sethi [17]; Lenstra, Kan y Brucker [53]). Cuando m es parte de la entrada, los problemas son fuertemente NP-hard, independiente del ambiente de máquinas que se considere (Garey y Johnson [36], [37]; Horowitz y Sahni [46], [47]; Lawler et al. [52]).

1.4. Algoritmos de aproximación

Muchos problemas de optimización de interés práctico son intratables. Por lo tanto, una forma práctica de resolver estos problemas, es usando algoritmos de aproximación. Los que permiten encontrar soluciones cercanas al óptimo, en una cantidad razonable de tiempo de cómputo. La eficiencia de estos métodos es evaluada por dos aspectos principales: la calidad de la solución obtenida y el tiempo necesario para obtener tal solución. La complejidad de los problemas y la existencia de este trade-off entre calidad y tiempo de ejecución, hace deseable trabajar con algoritmos que corran en tiempo acotado por un polinomio con respecto al tamaño de la instancia. Los algoritmos de aproximación son algoritmos eficientes, que a pesar de que no encuentran necesariamente la solución óptima del problema, permiten encontrar una solución cercana al óptimo con una certificación de su calidad.

Definición 1.5 *Un **algoritmo α-aproximación** para un problema de optimización es un algoritmo que corre en tiempo polinomial, que para todas las instancias del problema produce una solución cuyo valor esta alejando a lo más en una factor α con respecto al óptimo. El factor α es llamado **factor de aproximación** o **garantía de aproximación**. Cuando α no depende del tamaño de la instancia se dice que el algoritmo de aproximación es de **factor constante**.*

Definición 1.6 *Dado un problema de minimización, un **esquema de aproximación a tiempo polinomial (PTAS)** es una colección de algoritmos $\{A_\varepsilon\}_{\varepsilon>0}$ tal que A_ε es una $(1+\varepsilon)$-aproximación. Es importante notar que ε no es parte del input del problema, y por lo tanto el tiempo de ejecución podría ser exponencial en $1/\varepsilon$. Cuando el tiempo de ejecución es también polinomial en $1/\varepsilon$ decimos que $\{A_\varepsilon\}_{\varepsilon>0}$ es un **esquema de aproximación a tiempo completamente polinomial (FPTAS)**.*

Se sabe que a menos que P = NP, un problema de optimización fuertemente NP-hard no puede tener un FPTAS (Garey y Johnson, [37]).

Por lo general, los métodos de aproximación (heurísticos) son clasificados en: heurísticas de mejoramiento y heurísticas de construcción. Las heurísticas de mejoramiento, generalmente parten con una solución factible e iterativamente intentan obtener una mejor solución (en la sección 1.5 serán presentadas con mayor detalle). Una heurística de construcción, construye una solución asignando valores a una o varias variables a la vez. Un ejemplo de este tipo de heurísticas para problemas de programación de la producción son las Listas de Programación. También llamadas lista de tareas, lista de prioridad o list scheduling.

Definición 1.7 ***Lista de Programación*** *(Adaptado desde Graham [40]).*
Es una regla de prioridad establecida para las tareas. La cual se puede ver como una lista

ordenada de tareas de acuerdo a una regla de prioridad decreciente. En cualquier instante en que una máquina k completa una tarea (queda desocupada), se busca la siguiente tarea en la lista y esta es asignada a la máquina k.

Para el problema $P||C_{max}$, el factor de aproximación de cualquier lista de programación es $2-\frac{1}{m}$ (Graham, [39]). Si la lista se construye usando la regla LPT[3] el factor de aproximación mejora a $\frac{4}{3}-\frac{1}{3m}$ (Graham, [40]). Resolver este problema con una lista de programación requiere un tiempo de ejecución $O(n\log n)$. Hochbaum y Shmoys [42] proponen un PTAS cuyo tiempo de ejecución es $O((n/\varepsilon)^{1/\varepsilon^2})$. También proponen versiones más prácticas para $\varepsilon = 1/5+2^{-k}$ y $\varepsilon = 1/6+2^{-k}$, con tiempos de ejecución $O(n(k+\log n))$ y $O(n(km^4+\log n)$ respectivamente, donde k es la cantidad de iteraciones. Para m fijo, Sahni [61] propone un FPTAS cuyo tiempo de ejecución es $O(n/\varepsilon^2)$.

Para el problema $Q||C_{\text{máx}}$, Gonzalez, Ibarra y Sahni [38], propusieron una generalización de la regla LPT, la que permite alcanzar soluciones con un factor de aproximación $\frac{2m}{m+1}$, con una cota inferior de 1,5. Posteriormente, Cho y Sahni [20], determinan las cotas superiores $1+\frac{\sqrt{2m-2}}{2}$ para $m>2$, y 1,618 para $m=2$. En Dobson [26], se ajustan las cotas sobre el factor de aproximación, determinando que el factor de aproximación se encuentra entre 1,51 y 1,58. Posteriormente, en Friesen [32], se establece que el factor se encuentra en el rango 1,52 y 1,67. En Friesen y Langston [33], se presenta un algoritmo de aproximación (MULTIFIT) con una cota superior para el factor de aproximación de 1,4, cuyo tiempo de ejecución es $O(n\log n+kn\log m)$. Posteriormente, Chen [19] determina que este factor se encuentra entre 1,36 y 1,82. En Hochbaum y Shmoys [43], se presenta un PTAS con tiempo de ejecución $O((m/\varepsilon)(n/\varepsilon)^{1/\varepsilon^2})$. Posteriormente, en Epstein y Sgall [29] se presenta una generalización de este resultado. Para m fijo, Horowitz y Sahni [46], proponen un FPTAS con tiempo de ejecución $O(nm(nm/\varepsilon)^{m-1})$.

Para el problema $R||C_{max}$, Davis y Jaffe [25] presentan un algoritmo basado en una lista de programación, que tiene un factor de aproximación $2\sqrt{m}$, y puede ser ejecutado en $O(nm\log n)$. También se propone un algoritmo con factor de aproximación $1{,}5\sqrt{m}$, cuyo tiempo de ejecución es $O(m^m+mn\log n)$. En los trabajos de Garey y Johnson ([35] y [36]), se concluye que no existe un FPTAS para el problema $R||C_{max}$ (incluso para el problema $Q||C_{max}$). En Lenstra, Shmoys y Tardos [54] se presenta un 2-algortimo de aproximación cuyo tiempo de ejecución es $O((n+1)^{m/\varepsilon})$, y demuestran que no existe un 3/2-algoritmo de aproximación en tiempo polinomial a menos que N=NP. Para m fijo, Horowitz y Sahni [46], proponen un FPTAS con tiempo de ejecución $O(nm(nm/\varepsilon)^{m-1})$. Más tarde, Jansen y Porkolab [50], proponen otro FPTAS con tiempo de ejecución $O(n(m/\varepsilon)^{O(m)})$. Este algoritmo consiste en separar las tareas en dos grupos de acuerdo a sus tiempos de proceso, luego se combinan los algoritmos propuestos en Horowitz y Sahni [46] y Lenstra et al. [54].

Para el problema $P_m||\sum w_jC_j$, Sahni [61] presenta un FPTAS con tiempo de ejecución $O(n(n^2/\varepsilon)^{m-1})$. Kawaguchi y Kyan [51] analizan la lista de programación en orden decreciente de los cocientes de Smith (w_j/p_j), y demuestran un factor de aproximación constante $\frac{1}{2}(1+\sqrt{2})$. Skutella y Woeginger [66] proponen un PTAS, basado en agrupar tareas con similar cociente de Smith. Para el problema $Q_m||\sum w_jC_j$, Horowitz y Sahni [46] proponen un FPTAS con tiempo de ejecución $O(n(n^2/\varepsilon)^{m-1})$. Para el problema $R||\sum w_jC_j$, Schulz y

[3]Longest processing time first

Skutella [62] proponen un algoritmo basado en programación lineal con factor de aproximación $3/2+\varepsilon$. Mientras que en Skutella [65], se presenta un algoritmo que usa redondeo sobre una relajación cuadrática del problema. Este algoritmo exhibe un factor de aproximación de 3/2. Recientemente Bansal, Srinivasan y Svensson [10] mejoran este resultado, modificando la técnica de redondeo, obteniendo un $(3/2-c)$-PTAS.

En Leung y Li [55], se presenta una revisión de algoritmos de aproximación para distintos problemas en ambientes de máquinas paralelas restringidas, donde el objetivo es minimizar el makespan.

1.5. Búsqueda local

La literatura dedicada a algoritmos de aproximación distingue dos clases de algoritmos: algoritmos de construcción, y algoritmos de mejoramiento. Un algoritmo de construcción, construye una solución asignando valores a una o varias variables a la vez. Un algoritmo de mejoramiento, también llamado algoritmo de búsqueda local o búsqueda en vecindario, generalmente parte con una solución factible e iterativamente intenta obtener una mejor solución. Para una introducción a la búsqueda local se recomiendan los textos de Aarts y Lenstra [1], y Michiels, Aarts y Korst [56].

La forma más simple de búsqueda local es el mejoramiento iterativo. Este método, parte con una solución inicial factible, la cual es modificada iterativamente. Estas modificaciones sucesivas mejoran el costo de la solución. Las modificaciones se realizan de acuerdo a una relación de adyacencia, definida por una estructura de vecindario preestablecida. El procedimiento se detiene cuando no se encuentran mejores soluciones adyacentes o vecinas. La solución obtenida de esta forma, es un óptimo local para el vecindario utilizado.

Para utilizar búsqueda local, es necesario definir la estructura de vecindario. La que permite determinar las soluciones vecinas que pueden ser alcanzadas al realizar una movida. A continuación se describen los aspectos más relevantes relacionados con búsqueda local.

Sea $\mathcal{I}$ una instancia del problema $\mathcal{P}$, $\mathcal{S}$ el conjunto de soluciones factibles de $\mathcal{I}$, $\boldsymbol{s} \in \mathcal{S}$ una solución factible de $\mathcal{I}$, $C(\boldsymbol{s})$ el costo de la solución $\boldsymbol{s}$.

Definición 1.8 ***Estructura de vecindario-ν.***
Procedimiento que define los cambios que se deben realizar sobre una solución $\boldsymbol{s} \in \mathcal{S}$, para determinar las soluciones adyacentes a $\boldsymbol{s}$. Sea $\dot{\boldsymbol{s}}$ alguna solución adyacente obtenida al usar la estructura de vecindario ν. Se dice que la solución $\dot{\boldsymbol{s}}$ es un ***ν-vecino*** *de $\boldsymbol{s}$.*

Definición 1.9 ***ν-movida.***
Procedimiento que se realiza sobre la solución $\boldsymbol{s} \in \mathcal{S}$ para obtener una determinada solución ν-vecina de $\boldsymbol{s}$.

Definición 1.10 ***Conjunto de soluciones ν-vecinas.***
$\mathcal{N}(\boldsymbol{s})$, el conjunto de todas las soluciones que pueden ser obtenidas al aplicar una ν-movida

sobre la solución $\boldsymbol{s}$.

Definición 1.11 ***Conjunto de soluciones ν-candidatas.***
$\mathcal{N}'(\boldsymbol{s}) \subseteq \mathcal{N}(\boldsymbol{s})$, *el conjunto de todas las soluciones ν-vecinas de* $\boldsymbol{s}$, *que tienen mejor costo que la solución* $\boldsymbol{s}$.
Para un problema de minimización se tiene que, $\mathcal{N}'(\boldsymbol{s}) = \{\dot{\boldsymbol{s}} \in \mathcal{N}(\boldsymbol{s}) : \ C(\dot{\boldsymbol{s}}) < C(\boldsymbol{s})\}$.
En cambio, para un problema de maximización, $\mathcal{N}'(\boldsymbol{s}) = \{\dot{\boldsymbol{s}} \in \mathcal{N}(\boldsymbol{s}) : \ C(\dot{\boldsymbol{s}}) > C(\boldsymbol{s})\}$.

Cuando se utiliza búsqueda local, se pueden definir diversos criterios de detención para el proceso iterativo. Uno de estos criterios, consiste en detener el proceso de búsqueda cuando no hay soluciones vecinas con mejor costo. La solución obtenida bajo este criterio de detención, es un óptimo local para la estructura de vecindario usada.

Definición 1.12 ***Solución ν-óptima.***
La solución $\boldsymbol{s}^{\nu} \in \boldsymbol{S}$ *es un* ***ν-óptimo*** *de la instancia* $\mathcal{I}$, *si no existen soluciones ν-vecinas a* $\boldsymbol{s}^{\nu}$ *con un mejor costo. Esto es* $|\mathcal{N}'(\boldsymbol{s}^{\nu})| = 0$. *Luego, en un problema de minimización se tiene que* $C(\boldsymbol{s}^{\nu}) \leq C(\boldsymbol{s})$, $\forall \boldsymbol{s} \in \mathcal{N}(\boldsymbol{s}^{\nu})$. *Mientras que en un problema de maximización,* $C(\boldsymbol{s}^{\nu}) \geq C(\boldsymbol{s})$, $\forall \boldsymbol{s} \in \mathcal{N}(\boldsymbol{s}^{\nu})$.

A continuación se presenta un algoritmo genérico de búsqueda local, el cual entrega un óptimo local.

Algoritmo Búsqueda Local

1: Sea $\boldsymbol{s} \in \mathcal{S}$ una solución inicial factible
2: Sea $\mathcal{N}(\boldsymbol{s})$ el conjunto de soluciones vecinas de $\boldsymbol{s}$
3: Sea $\mathcal{N}'(\boldsymbol{s}) \subseteq \mathcal{N}(\boldsymbol{s})$ el conjunto de soluciones candidatas en $\mathcal{N}(\boldsymbol{s})$
4: **while** $|\mathcal{N}'(\boldsymbol{s})| > 0$ **do**
5: Seleccionar una solución $\boldsymbol{z} \in \mathcal{N}'(\boldsymbol{s})$
6: $\boldsymbol{s} \leftarrow \boldsymbol{z}$
7: **return** $\boldsymbol{s}$

De acuerdo a lo planteado por Ahuja et al. [4], hay dos aspectos muy importantes que determinan la eficiencia de un algoritmo de búsqueda local: tamaño del vecindario y calidad de los óptimos locales. Una forma de evaluar la calidad de los óptimos locales es el análisis de peor caso. En Angel [5] se presenta un revisión de análisis de peor caso y otros aspectos teóricos de búsqueda local, donde se presentan resultados para una gran variedad de problemas, incluyendo problemas de programación de tareas. En la sección 1.6 se presentan algunas estructuras de vecindario utilizadas en programación de tareas en máquinas paralelas.

1.6. Vecindarios

El desempeño de la estructura de vecindario seleccionada depende del problema que se intenta resolver. De acuerdo a lo planteado en Aarts y Lenstra [1], encontrar vecindarios eficientes que permitan obtener óptimos locales de alta calidad es un desafío.

Los vecindarios se pueden clasificar de acuerdo a su tamaño. Un vecindario de búsqueda local se dice que es de tamaño polinomial, si la cantidad de vecinos para cada solución factible es polinomialmente acotada por el tamaño de la entrada. En cambio, es de tamaño exponencial si la cantidad de vecinos para algunas soluciones crece al menos exponencialmente con el tamaño de la entrada. En Ahuja et al. [4], se analiza el desempeño de vecindarios de gran tamaño con respecto al tamaño de la entrada de datos. Donde es posible realizar una búsqueda en el vecindario de manera eficiente.

En Brueggemann et al. [16] se presentan resultados para el problema $P||C_{max}$, donde se compara el desempeño de vecindarios de tamaño polinomial, y exponencial que permiten búsquedas en tiempo polinomial. Schuurman y Vredeveld [63] estudian el desempeño de vecindarios de tamaño polinomial para los problemas $P, Q, R||C_{max}$. Los vecindarios estudiados son jump, swap y push.

En la tabla 1.2 se presentan resultados para variados problemas de máquinas paralelas, para distintos vecindarios. Ahí se puede apreciar que el objetivo de minimizar el makespan (C_{max}) a sido ampliamente estudiado. En cambio, objetivos como $\sum w_j C_j$ y $\sum C_j$ han sido escasamente estudiados.

En lo que sigue de esta sección se usará la siguiente notación. Sea $\mathcal{I}$ una instancia del problema $\mathcal{P}$, $\mathcal{S}$ el conjunto de soluciones factibles de $\mathcal{I}$, $\mathcal{M}$ el conjunto de máquinas, $\mathcal{J}$ el conjunto de tareas, $\boldsymbol{s} \in \mathcal{S}$ alguna solución de $\mathcal{I}$, $C(\boldsymbol{s})$ el costo de la solución $\boldsymbol{s}$, $\mathcal{J}_i(\boldsymbol{s})$ el conjunto de tareas asignadas a la máquina $i \in \mathcal{M}$ en la solución $\boldsymbol{s}$.

1.6.1. Vecindario de Jump

La movida de Jump, también llamada inserción, salto o move. Consiste en reasignar una tarea a otra máquina. Esto es, sacar una tarea de una máquina y asignarla a otra máquina factible. Jump es un vecindario de tamaño polinomial. En la figura 1.1 se presenta un esquema representativo de una movida de jump.

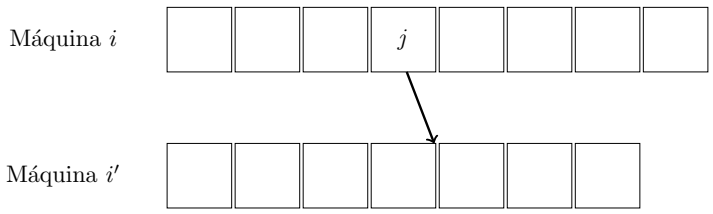


Figura 1.1: Movida de Inserción/Jump

Los aspectos más importantes de este vecindario se definen a continuación.

Definición 1.13 *Una **movida de jump** es un procedimiento, que al ser aplicado sobre la solución $\boldsymbol{s} \in \mathcal{S}$, permite determinar una solución **jump-vecina** de $\boldsymbol{s}$. Sea $\boldsymbol{s}_{ijt} \in \mathcal{S}$ la solución jump-vecina de $\boldsymbol{s}$, que se obtiene al mover la tarea $j \in \mathcal{J}_i(\boldsymbol{s})$ desde la máquina $i \in \mathcal{M}$ a la máquina $t \in \mathcal{M} \setminus \{i\}$.*

Definición 1.14 ***Conjunto de soluciones jump-vecinas.***
$\mathcal{N}(\boldsymbol{s}) = \{\boldsymbol{s}_{ijt} \in \mathcal{S} : \ i \in \mathcal{M},\ t \in \mathcal{M} \setminus \{i\},\ j \in \mathcal{J}_i(\boldsymbol{s})\}$.
El conjunto de soluciones jump-vecinas de $\boldsymbol{s}$*, es el conjunto de soluciones factibles que se obtiene al realizar alguna movida de jump sobre la solución* $\boldsymbol{s}$*.*

Definición 1.15 ***Candidatos del vecindario de jump.***
$\mathcal{N}'(\boldsymbol{s}) = \{\boldsymbol{z} \in \mathcal{N}(\boldsymbol{s}) : \ C(\boldsymbol{s}) < C(\boldsymbol{z})\}$.
El conjunto de soluciones candidatas, es el conjunto de soluciones jump-vecinas de $\boldsymbol{s}$*, que tienen un costo estrictamente mejor que el de la solución* $\boldsymbol{s}$*.*

Definición 1.16 ***Solución jump-óptima.***
$\boldsymbol{s} \in \mathcal{S}$ *es una solución jump-óptima, si y solo si* $\mathcal{N}'(\boldsymbol{s}) = \varnothing$*. Esto es, no existen soluciones candidatas para* $\boldsymbol{s}$*.*

1.6.2. Vecindario de Swap

La movida de Swap, también llamada intercambio, consiste en intercambiar dos tareas. Esto es, seleccionar dos tareas asignadas a diferente máquina, e intercambiar la máquina asignada entre estas dos tareas. Swap es un vecindario de tamaño polinomial. En la figura 1.2 se presenta un esquema representativo de una movida de swap.

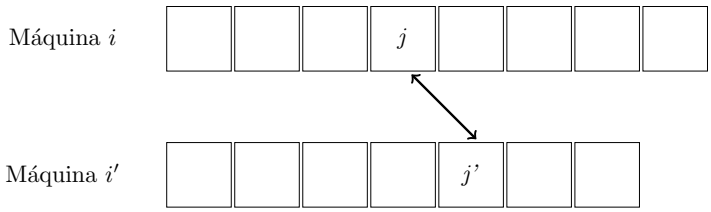


Figura 1.2: Movida de Intercambio/Swap

Los aspectos más importantes de este vecindario se definen a continuación.

Definición 1.17 *Una* ***movida de swap*** *es un procedimiento, que al ser aplicado sobre la solución* $\boldsymbol{s} \in \mathcal{S}$*, permite determinar una solución* ***swap-vecina*** *de* $\boldsymbol{s}$*. La movida consiste en seleccionar dos máquinas distintas* $i, t \in \mathcal{M}$*. Se selecciona una tarea* $j \in \mathcal{J}_i(\boldsymbol{s})$ *asignada a la máquina* i*; y una tarea* $k \in \mathcal{J}_t(\boldsymbol{s})$ *asignada a la máquina* t*, o bien, no se selecciona ninguna tarea de la máquina* t*. La tarea* j *es movida a la máquina* t *y la tarea* k *es movida a la máquina* i*. Sea* $\boldsymbol{s}_{ijtk} \in \mathcal{S}$ *la solución swap-vecina obtenida.*

De acuerdo a la definición de movida de swap, se tiene que las movidas de jump están incluidas en el vecindario de swap. Luego, el conjunto de soluciones jump-vecinas de cualquier solución $\boldsymbol{s} \in \mathcal{S}$ esta contenido completamente en el conjunto de soluciones swap-vecinas $\boldsymbol{s} \in \mathcal{S}$.

Definición 1.18 ***Conjunto de soluciones swap-vecinas.***
$\mathcal{N}(\boldsymbol{s}) = \{\boldsymbol{s}_{ijt} \in \mathcal{S} : \ i \in \mathcal{M},\ t \in \mathcal{M} \setminus \{i\},\ j \in \mathcal{J}_i(\boldsymbol{s}),\ k \in \mathcal{J}_t(\boldsymbol{s}) \cup \varnothing\}$.

El conjunto de soluciones swap-vecinas de $\boldsymbol{s}$, es el conjunto de soluciones factibles que se obtiene al realizar una movida de swap sobre la solución $\boldsymbol{s}$.

Definición 1.19 ***Candidatos del vecindario de swap.***
$\mathcal{N}'(\boldsymbol{s}) = \{\boldsymbol{z} \in \mathcal{N}(\boldsymbol{s}) : \ C(\boldsymbol{s}) < C(\boldsymbol{z})\}$.
El conjunto de soluciones candidatas, es el conjunto de soluciones swap-vecinas de $\boldsymbol{s}$, que tienen un costo estrictamente mejor que el de la solución $\boldsymbol{s}$.

Definición 1.20 ***Solución swap-óptima.***
$\boldsymbol{s} \in \mathcal{S}$ es una solución swap-óptima, si y solo si $\mathcal{N}'(\boldsymbol{s}) = \varnothing$. Esto es, no existen soluciones candidatas para $\boldsymbol{s}$.

1.6.3. Vecindario de Exjump

La movida de exjump se define de igual manera que la movida de jump. Luego, el conjunto de soluciones exjump-vecinas de una solución $\boldsymbol{s} \in \mathcal{S}$ es idéntico al conjunto de soluciones jump-vecinas de $\boldsymbol{s}$, por lo tanto es un vecindario de tamaño polinomial. La diferencia entre estos vecindarios es la construcción del conjunto de soluciones candidatas. Para el vecindario de jump, este conjunto está compuesto por todas las soluciones con costo estrictamente mejor que $C(\boldsymbol{s})$. Mientras que el conjunto de soluciones candidatas para el vecindario de exjump está compuesto por las soluciones que tienen igual o mejor costo que $C(\boldsymbol{s})$.

Definición 1.21 ***Candidatos del vecindario de exjump.***
$\mathcal{N}'(\boldsymbol{s}) = \{\boldsymbol{z} \in \mathcal{N}(\boldsymbol{s}) : \ C(\boldsymbol{s}) \leq C(\boldsymbol{z})\}$.
El conjunto de soluciones candidatas, es el conjunto de soluciones jump-vecinas de $\boldsymbol{s}$, que tienen un costo igual o mejor que el de la solución $\boldsymbol{s}$.

Definición 1.22 ***Solución exjump-óptima.***
$\boldsymbol{s} \in \mathcal{S}$ es una solución exjump-óptima, si y solo si $\mathcal{N}'(\boldsymbol{s}) = \varnothing$. Esto es, no existen soluciones candidatas para $\boldsymbol{s}$.

1.6.4. Otros vecindarios

A continuación se describen otras estructuras de vecindario, utilizadas para problemas de máquinas en paralelo donde el objetivo es la minimización del makespan. Sea $C_{max}(\boldsymbol{s})$ el makespan de la solución $\boldsymbol{s} \in \boldsymbol{S}$.

Definición 1.23 ***Máquina crítica.***
Es una máquina, cuya carga de trabajo es exactamente igual al makespan. Es decir es alguna de las máquinas que determinan el makespan para alguna solución $\boldsymbol{s} \in \boldsymbol{S}$.

Vecindario de Exjump

Esta estructura de vecindario de tamaño polinomial, es una extensión del vecindario de jump. Este vecindario es utilizado en problemas de máquinas en paralelo, donde el objetivo es

la minimización del makespan (Brueggemann et al. [14], [16]; Czumaj y Vöcking [24]; Recalde et al. [59]; Rutten et al. [60]; Schuurman y Vredeveld [63]; Vöcking [68]). Para su aplicación, las máquinas son ordenadas en forma decreciente con respecto a la carga de trabajo que tienen asignadas. Una movida de lexjump reduce lexicográficamente el vector ordenado de cargas.

Vecindario de Push

Esta estructura de vecindario de tamaño polinomial, fue introducida por Schuurman y Vredeveld [63]. Fue utilizada para los problemas $P||C_{max}$ y $Q||C_{max}$. Recalde et al. [59] usó este vecindario para los problemas $RP||C_{max}$ y $RQ||C_{max}$. Una movida de push consta de una secuencia de movidas de jump. A partir de un solución $\boldsymbol{s} \in \mathcal{S}$, con makespan $C_{max}(\boldsymbol{s})$, se selecciona una tarea k asignada a una máquina crítica, y se selecciona una máquina no crítica i para asignar la tarea k. Se dice que k encaja en i si $p_{ik} + \sum_{j \in \mathcal{J}_i(\boldsymbol{s}):\ p_{ij} \geq p_{ik}} p_{ij} < C_{max}(\boldsymbol{s})$. Si la tarea k no encaja en ninguna máquina, entonces la tarea k no puede ser movida (empujada). Si después de mover la tarea k a la máquina i, la carga de la máquina i es mayor o igual al makespan original, iterativamente se remueve la tarea con menor tiempo de proceso en la máquina i. Esto se repite hasta que la carga de trabajo en la máquina i sea menor a $C_{max}(\boldsymbol{s})$. Las tareas removidas generan una cola de tareas pendientes. Las que son reasignadas de acuerdo a una prioridad establecida por la regla LPT. De acuerdo a la prioridad establecida, se realizan movidas de push para las tareas. Si la primera tarea en la lista remanente no encaja en ninguna máquina, entonces no existen soluciones vecinas. Se alcanza una solución localmente óptima cuando ninguna tarea asignada a las máquinas críticas puede ser movida. Durante una movida de push, a lo más n tareas necesitan ser movidas, y una movida puede ser realizada en $O(n)$. Entonces una movida de push requiere $O(n^2)$ operaciones. De acuerdo a lo planteado por Schuurman y Vredeveld [63], si se usa una estructura de datos apropiada, y se hacen otras consideraciones, una movida de push puede ser realizada en $O(n \log n)$.

Vecindario de Split

Esta estructura de vecindario de tamaño exponencial, fue introducida por Brueggemann et al. [14], [16], y fue utilizada para el problema $P||C_{max}$. También fue usada en conjunto con vecindarios de jump y lexjump. Para usar esta estructura de vecindario, se requiere un operador de split (partición). Para cada máquina $i \in \mathcal{M}$, el conjunto $\mathcal{J}_i(\boldsymbol{s})$ de tareas asignadas a la máquina i, se divide en dos conjuntos disjuntos $\mathcal{J}_i^1(\boldsymbol{s})$ y $\mathcal{J}_i^2(\boldsymbol{s})$, generando $2m$ conjuntos (2 por cada máquina). Luego, los conjuntos $\mathcal{J}_i^1(\boldsymbol{s})$ y $\mathcal{J}_i^2(\boldsymbol{s})$ se asignan en diferentes máquinas. La solución vecina seleccionada tiene estrictamente un menor makespan, o el mismo makespan pero con una menor cantidad de máquinas críticas. La cantidad de vecinos en cada solución factible es $(2m)!/(2^m m!)$, y en cada movida el mejor vecino puede ser determinado en $O(m \log m)$.

Vecindario de Multi-intercambio

Esta estructura de vecindario de tamaño exponencial, fue introducida por Frangioni, Necciari y Scutella [31] para el problema $R||C_{max}$, y fue utilizada para los problemas $P, Q, R||C_{max}$ por Schuurman y Vredeveld [63]. Una movida de multi-intercambio consiste de una secuencia de movidas de swap, donde no necesariamente dos máquinas intercambian tareas. Los vecindarios de jump y swap son casos especiales de esta estructura de vecindario. A diferencia de los vecindarios de jump y swap, la cantidad de soluciones vecinas puede crecer exponencialmente con respecto al tamaño del problema. Más aún, Frangioni et al. [31] mostró que

decidir si existe una movida de mejora es un problema NP-completo. Se dice que la solución $\boldsymbol{s}$ es un óptimo local para este vecindario, si no existen movidas que disminuyan el makespan o la cantidad de máquinas críticas.

1.6.5. Tiempos de corrida

Por lo general, para estructuras de vecindario de tamaño polinomial se tienen tiempos de corrida polinomiales. Para el problema $P||C_{max}$, Finn y Horowitz [30], afirmaron que en el peor caso se requieren $2n-1$ movidas para encontrar un jump-óptimo. Sin embrago, este resultado fue corregido por Hurkens y Vredeveld [48] a $O(n^2)$. Quienes también plantean que para el problema $Q||C_{max}$ se puede alcanzar un jump-óptimo en $O(nm)$ iteraciones. Para el problema $R||C_{max}$, Piersma y van Dijk [57], muestran que al usar un vecindario de jump o swap, cada búsqueda en el vecindario tiene un tiempo de corrida $O(n^2m^2)$.

Brucker, Hurink y Werner ([12], [13]), realizan un análisis de peor caso para búsqueda local con vecindario de jump y lexjump. Determinan que la cantidad de movidas requeridas para encontrar un jump-óptimo para el problema $P_m||C_{max}$ es a lo más $O(n^2)$. Para $m=2$ este resultado puede ser mejorado a $O(n)$, si se selecciona la mejor movida en todas las iteraciones. Este resultado también fue presentado por Hurkens y Vredeveld [48] para el problema $Q_2||C_{max}$.

En Abed [2] y Abed, Correa y Huang [3], se determina que para el problema $R||\sum w_jC_j$ se puede determinar un óptimo local en tiempo polinomial . Con la consideración de que en cada iteración se selecciona la movida que reduce en mayor cantidad el costo de la solución.

1.7. Aproximación por búsqueda local

En esta sección se presenta una revisión de resultados relacionados con la calidad de los óptimos locales para diversos vecindarios. Se utilizará la siguiente notación: m, cantidad de máquinas; n, cantidad de tareas; s_i: velocidad de la máquina i. Para este útlimo parámetro, se considera que la velocidad mínima es unitaria, es decir $s_{\text{mín}} = \text{mín}_{i\in\mathcal{M}}\{s_i\} = 1$. Además se define $s_{\text{máx}} = \text{máx}_{i\in\mathcal{M}}\{s_i\}$, y $S = \sum_{i\in\mathcal{M}} s_i$. En la tabla 1.2 se resume la información recopilada.

Para el problema $P||C_{max}$, Finn y Horowitz [30] determinaron la cota superior $2-\frac{2}{m+1}$ para el vecindario jump. Posteriormente, Schuurman y Vredeveld [63] determinaron que la cota era ajustada e incluyeron otros vecindarios que permitían alcanzar el mismo factor de aproximación, swap, lexjump y multi-intercambio. Para el vecindario de push, determinan que el factor de aproximación se encuentra entre $\frac{4m}{3m+1}$ y $\frac{4}{3}-\frac{1}{3m}$. Para el caso particular de dos máquinas en paralelo determinan un factor de aproximación ajustado de $\frac{8}{7}$. Brueggemann et al. [16] determinaron que para el vecindario de split el factor de aproximación es $2-\frac{2}{m+1}$. Este factor mejora a $2-\frac{4}{m+3}$ si se incluye el vecindario jump. Si se incluye el vecindario lexjump, el factor de aproximación se encuentra entre $\frac{3}{2}$ y $\frac{3}{2}-\frac{1}{2m-2}$. Por otra parte, Vöcking

Problema	Factor de aproximación	Vecindario	Ref.
$P\|\|C_{max}$	$2-\frac{2}{m+1}$	jump	[30], [63]
$P\|\|C_{max}$	$2-\frac{2}{m+1}$	swap, me, lexjump	[63]
$P\|\|C_{max}$	$LB=\frac{4m}{3m+1}$; $UB=\frac{4}{3}-\frac{1}{3m}$	push	[63]
$P_2\|\|C_{max}$	$\frac{8}{7}$	push	[63]
$P\|\|C_{max}$	$2-\frac{2}{m+1}$	split	[16]
$P\|\|C_{max}$	$2-\frac{2}{m+1}$	lexjump	[68]
$P\|\|C_{max}$	$2-\frac{4}{m+3}$	split+jump	[16]
$P\|\|C_{max}$	$LB^a=\frac{3}{2}-\frac{1}{2m-2}$; $UB^b=\frac{3}{2}$	split+lexjump	[16]
$P\|\|\sum w_jC_j$	$LB=\frac{6}{5}$; $UB=\frac{3}{2}-\frac{1}{2m}$	jump	[15]
$Q\|\|C_{max}$	$\frac{1+\sqrt{4m-3}}{2}$	jump	[20]
$Q\|\|C_{max}$	$\frac{1+\sqrt{4m-3}}{2}$	swap, me	[63]
$Q\|\|C_{max}$	$LB=\frac{3}{2}-\varepsilon$; $UB=2-\frac{2}{m+1}$	push	[63]
$Q_2\|\|C_{max}$	$\frac{\sqrt{17}+1}{4}$	push	[63]
$Q\|\|C_{max}$	$\Theta(\frac{\log m}{\log\log m})$	lexjump	[24], [68]
$R\|\|C_{max}$	$LB=\frac{p_{max}}{C^*_{max}}$ [c]	jump, swap, me	[63]
$R_2\|\|C_{max}$	$LB=\frac{p_{max}}{C^*_{max}}$ [d]	jump	[63]
$R_2\|\|C_{max}$	$LB=n-1$	swap, me	[63]
$R_2\|\|C_{max}$	$\frac{1+\sqrt{5}}{2}$	jump[e]	[49]
$RP\|\|C_{max}$	$\frac{1}{2}+\sqrt{m-\frac{3}{4}}$	jump, swap, push	[59], [60]
$RP\|\|C_{max}$	$\Theta\left(\frac{\log m}{\log\log m}\right)$	jump	[6], [34]
$RQ\|\|C_{max}$	$UB=\frac{1}{2}+\sqrt{\frac{1}{4}+(m-1)s_{max}}$	jump, swap, push	[59], [60]
$RQ\|\|C_{max}$	$LB=\sqrt{\frac{1}{4}+(m-1)s_{max}}$	jump, swap, push	[59]
$RQ\|p_j=1\|C_{max}$	$\sqrt{\left(1+\frac{m-1}{n}\right)S}$	jump, swap, push	[59], [60]
$RQ\|\|C_{max}$	$O(\frac{\log S}{\log\log S})$[f]	lexjump	[59], [60]
$RQ\|\|C_{max}$	$LB=\sqrt{(m-2)s_{max}}$ [g]	jump, swap, push	[60]

[a] LB: Cota inferior.
[b] UB: Cota superior.
[c] $p_{max}=max_{j\in\mathcal{J}}\{p_j\}$, C^*_{max}: makespan óptimo
[d] Ver c
[e] Se utiliza una solución inicial, definida por los tiempos de proceso.
[f] Para $S\geq 16$ tiene convergencia asintótica.
[g] Para $m>2,\ s_{max}\geq 2$.

Tabla 1.2: Aproximaciones por búsqueda local

[68] estudia el problema descentralizado y determina el mismo factor de aproximación para los vecindarios de jump y lexjump. Para esta clase de problemas, se considera que las tareas son administradas por agentes que toman la decisión de dónde asignar la tarea. Esta decisión es tomada considerando su propio beneficio sin considerar el objetivo social. Para el caso de minimización del makespan, los agentes asignan su tarea a la máquina con menor carga de trabajo. Por esta vía se consigue alcanzar un equilibrio de Nash. Para evaluar el desempeño del juego se utiliza el precio de la anarquía.

Para el problema $P||\sum w_jC_j$, Brueggemann et al. [15] determinaron que el factor de aproximación de las soluciones jump-óptimas se encuentra entre $\frac{6}{5}$ y $\frac{3}{2}-\frac{1}{2m}$. Además, determinan un factor ajustado de $\frac{9-\sqrt{6}}{6}$ para el caso especial donde todas las tareas tienen el mismo cociente de Smith .

Para el problema $Q||C_{max}$, Cho y Sahni [20] establecen la cota superior $\frac{1+\sqrt{4m-3}}{2}$ para el vecindario de jump. Posteriormente, Schuurman y Vredeveld [63] demuestran que la cota es ajustada y amplían el resultado a otros vecindarios, swap y multi-intercambio. Para el vecindario de push determinan que el factor de aproximación se encuentra entre $\frac{3}{2}-\varepsilon$ y $2-\frac{2}{m-1}$. Para el caso particular de $Q_2||C_{max}$ obtienen un factor de aproximación de $\frac{1+\sqrt{17}}{4}$. En Vöcking [68] se estudia una versión descentralizada de los problemas $P||C_{max}$ y $Q||C_{max}$. Se cuantifica la ineficiencia de los equilibrios de Nash. Para el problema $P||C_{max}$ el óptimo local se alcanza cuando cada agente es activado a los más una vez. Para estos problemas, la regla LPT permite construir un equilibrio puro. En Vöcking [68], y Czumaj y Vöcking [24] se determina que el factor de aproximación es $\Theta\left(\frac{\log m}{\log\log m}\right)$ para las soluciones lexjump-óptimas.

Para el problema $R||C_{max}$, Schuurman y Vredeveld [63] demuestran cotas inferiores para el factor de aproximación de las soluciones localmente óptimas para los vecindarios jump, swap y multi-intercambio. Para el caso especial $R_2||C_{max}$, Ibarra y Kim [49] demuestran un factor $\frac{1+\sqrt{5}}{2}$ para las soluciones jump-óptimas. Este resultado lo obtienen tomando como solución inicial, una construcción donde las tareas se asignan a la máquina que ofrece menor tiempo de proceso. El tiempo de ejecución para este algoritmo es $O(n\log n)$.

Para el problema $RP||C_{max}$, Recalde et al. [59] y Rutten et al. [60] determinan un factor de aproximación ajustado para los vecindarios de jump, swap y push. En Awerbuch et al. [6] y Gairing et al. [34] se estudio el problema de rutear tráfico en redes no-cooperativas, donde los enlaces son idénticos, y determinan un factor de aproximación para el peor caso con convergencia asintótica $\Theta\left(\frac{\log m}{\log\log m}\right)$. Este problema es equivalente a utilizar un vecindario de jump para el problema $RP||C_{max}$. Por otra parte, Czumaj y Vöcking [24] estudiaron el problema de rutear tráfico en redes no-cooperativas, donde los enlaces pueden tener diferentes velocidades, y determinan un factor de aproximación para el peor caso con convergencia asintótica $\Theta\left(\frac{\log m}{\log\log\log m}\right)$. Este problema es equivalente a utilizar un vecindario de jump para el problema $RQ||C_{max}$.

En Correa y Queyranne [23], se estudia el problema $RQ||\sum w_jC_j$ descentralizado. Determinan que el precio de la anarquía para estrategias puras es 2 para el caso de tareas unitarias en máquinas idénticas. Para el caso general determinan que el precio de la anarquía para estrategias mixtas es 4. Hoeksma y Uetz [44], estudian el problema $Q||\sum C_j$ descentralizado,

y determinan que el precio de la anarquía se encuentra entre 1,58 y 2.

De acuerdo a la revisión anterior, se puede concluir que el makespan a sido ampliamente estudiado. Sin embargo, el tiempo de completación ponderado no ha tenido mucha atención.

1.8. Contribuciones de este trabajo

En esta tesis se determinó el factor de aproximación para las soluciones localmente óptimas de diversos problemas de máquinas paralelas donde el objetivo es la minimización del tiempo de completación ponderado. Los vecindarios estudiados son jump, exjump y swap. Estos factores de aproximación, permiten medir la calidad de los óptimos locales desde la perspectiva de peor caso. En la tabla 1.3 se presentan los resultados obtenidos para cada problema estudiado.

Para los ambientes R y RQ, se determinó que el factor de aproximación de las soluciones jump, swap o exjump óptimas es 2 para el caso no ponderado, y 2,618 para el caso ponderado. En la figura 1.3 se presenta un mapa de los resultados.

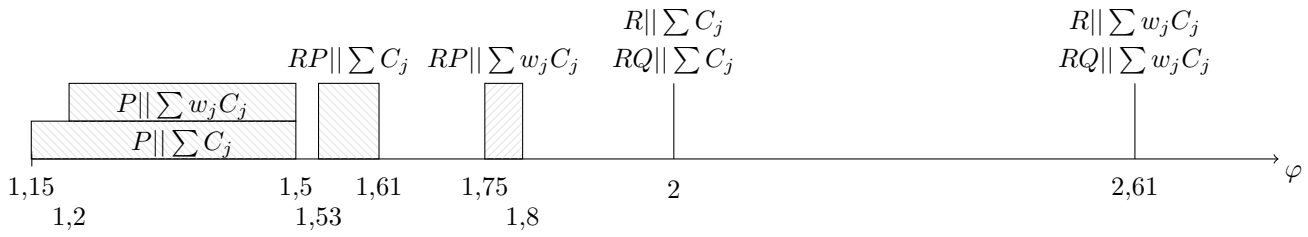


Figura 1.3: Mapa de resultados.

Para el ambiente RP, se estudiaron los vecindarios jump y swap. Se determinó un factor de aproximación ajustado para el problema $RP|p_j = 1|\sum C_j$. Para los problemas $RP||\sum C_j$ y $RP|p_j = 1|\sum w_jC_j$, se determinó que el factor de aproximación de una solución jump-óptima o swap-óptima se encuentra entre 1,533 y 1,618. Mientras que para el problema $RP||\sum w_jC_j$ estos factores de aproximación se encuentran entre 1,75 y 1,809.

Para el ambiente de máquinas paralelas idénticas P se presentan cotas superiores e inferiores para los factores de aproximación de las soluciones jump-óptimas. También se describen algunas características, que permiten identificar algunas instancias triviales. También se presentan factores de aproximación no constantes ajustados para las instancias de los problemas $P||\sum w_jC_j$ y $P||\sum C_j$.

Otra contribución de la tesis son las equivalencias entre problemas, demostradas en las secciones 3.5, 4.4 y 6.4. Donde, se demuestra que el problema $P|p_j = 1|\sum w_jC_j$ es equivalente al problema $P||\sum C_j$; el problema $RP|p_j = 1|\sum w_jC_j$ es equivalente al problema $RP||\sum C_j$; el problema $RQ|p_j = 1|\sum w_jC_j$ es equivalente al problema $RQ||\sum C_j$.

Problema	Factor de aproximación	Vecindario	Sección
$P\|p_j = 1\|\sum C_j$	$\varphi = 1$	jump	3.2
$P\|n \leq m+1\|\sum w_j C_j$	$\varphi = 1$	jump	3.2
$P\|\|\sum w_j C_j$	$\frac{6}{5} \leq \varphi \leq \frac{3}{2} - \frac{1}{2m}$	jump	3.3
$P\|\|\sum C_j$	$\frac{15}{13} \leq \varphi \leq \frac{3}{2} - \frac{1}{2m}$	jump	3.4
$P\|p_j = 1\|\sum w_j C_j$	$\frac{15}{13} \leq \varphi \leq \frac{3}{2} - \frac{1}{2m}$	jump	3.5
$RP\|\|\sum w_j C_j$	$1{,}75 \leq \varphi \leq 1{,}809$	jump, swap	4.2
$RP\|\|\sum C_j$	$1{,}533 \leq \varphi \leq 1{,}618$	jump, swap	4.3
$RP\|p_j = 1\|\sum w_j C_j$	$1{,}533 \leq \varphi \leq 1{,}618$	jump, swap	4.4
$RP\|p_j = 1\|\sum C_j$	$\varphi = 1{,}533$	jump, swap	4.5
$R\|\|\sum w_j C_j$	$\varphi = 2{,}618$	jump, swap, exjump	5.2
$R\|\|\sum C_j$	$\varphi = 2$	jump, swap, exjump	5.3
$RQ\|\|\sum w_j C_j$	$\varphi = 2{,}618$	jump, swap, exjump	6.2
$RQ\|\|\sum C_j$	$\varphi = 2$	jump, swap, exjump	6.3
$RQ\|p_j = 1\|\sum w_j C_j$	$\varphi = 2$	jump, swap, exjump	6.4
$RQ\|p_j = 1\|\sum C_j$	$\varphi = 2$	jump, swap, exjump	6.5

Tabla 1.3: Contribuciones de la Tesis

1.9. Organización de la Tesis

En el Capítulo 2 se presentan resultados preliminares, que serán usados para establecer cotas superiores e inferiores para los factores de aproximación. Estos factores de aproximación permitirán definir la calidad de los óptimos locales estudiados.

En el Capítulo 3 se presentan resultados para el ambiente de máquinas paralelas idénticas. Se incluyen factores de aproximación constantes y no constantes, que describen la calidad de las soluciones jump-óptimas. También se describen algunas características, que deben cumplir las instancias, para que el costo de cualquier solución jump-óptima sea distinto al costo del óptimo global.

En el Capítulo 4 se estudia la calidad de los óptimos locales, para vecindarios de jump y swap en el ambiente de máquinas paralelas idénticas restringidas.

En los Capítulos 5 y 6 se estudia la calidad de los óptimos locales para vecindarios de jump, exjump y swap. En el Capítulo 5 se presentan resultados para el ambiente de máquinas paralelas no-relacionadas. Finalmente, en el Capítulo 6 se presentan resultados para el ambiente de máquinas paralelas relacionadas restringidas.

En la tabla 1.3 se presentan los problemas estudiados y la sección asignada en esta Tesis.

Capítulo 2

Preliminares

En esta sección se presentan algunos resultados preliminares, que serán utilizados en los capítulos posteriores.

2.1. Trabajos relacionados

Complementando, la notación presentada en la sección 1.2, se define:

$\mathcal{I}$: una instancia del problema $\mathcal{P}$;

$\boldsymbol{x}^*$: alguna solución óptima de $\mathcal{I}$;

$\boldsymbol{x}$: alguna solución jump-óptima de $\mathcal{I}$;

$C(\boldsymbol{x}^*)$: costo óptimo de $\mathcal{I}$;

$C(\boldsymbol{x})$: costo del la solución jump-óptima $\boldsymbol{x}$ de $\mathcal{I}$.

2.1.1. Máquinas paralelas idénticas

Para el problema $P||\sum w_jC_j$, Eastman, Even e Isaacs [27] plantean el siguiente teorema, que permite acotar el costo de una solución óptima.

Teorema 2.1 *(Eastman, Even e Isaacs [27]).*
Para cualquier instancia del problema $P||\sum w_jC_j$, el costo de la solución óptima se encuentra acotada por:

$$\frac{\tilde{C}}{m} + \frac{(m-1)}{2m}\sum_{j\in\mathcal{J}} w_jp_j \leq C(\boldsymbol{x}^*) \leq \frac{\tilde{C}}{m} + \frac{(m-1)}{m}\sum_{j\in\mathcal{J}} w_jp_j \tag{2.1}$$

donde, $\tilde{C}$ representa el tiempo ponderado de completación si todas las tareas son asignadas en una máquina. Esto es:

$$\tilde{C} = \sum_{j \in \mathcal{J}} w_j \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \prec j}} p_k + \sum_{j \in \mathcal{J}} w_j p_j = \sum_{j \in \mathcal{J}} p_j \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} w_k + \sum_{j \in \mathcal{J}} w_j p_j$$

En Elmaghraby y Park [28], y Azizoglu y Kirca [7] se describen otras propiedades de las soluciones óptimas. Mientras que en Webster ([69], [70]) se presentan otras cotas inferiores para el problema $P||\sum w_j C_j$, incluyendo una variante del problema donde las tareas tienen tiempos de llegada (ready times) $P|r_j|\sum w_j C_j$. La cota inferior para el óptimo propuesta por Eastman et al. [27] es un caso particular de la cota propuesta por Webster [69]. Por otra parte, Kawaguchi y Kyan [51] evalúan el desempeño de la regla LRF (largest ratio first) que utiliza los conceptos presentados por Smith [67], construyendo una lista de scheduling en orden decreciente de $\frac{w_j}{p_j}$. La calidad de las soluciones construidas de esta manera tienen un factor de aproximación constante de $\frac{1}{2}\left(1+\sqrt{2}\right)$. Posteriormente, Schwiegelshohn [64] ofrece una demostración alternativa más simple.

Por otra parte, Brueggemann, Hurink y Kern [15] estudian la calidad de las soluciones jump-óptimas. En la sección 3.3, se presenta una demostración alternativa a la presentada por estos autores.

Teorema 2.2 *(Brueggemann, Hurink y Kern [15]).*
El factor de aproximación de una solución jump-óptima para el problema $P||\sum w_j C_j$, es a lo más $\frac{3}{2} - \frac{1}{2m}$.

Para el caso especial, donde todas las tareas tienen el mismo cociente de Smith, Brueggemann et al. [15] determinan que el factor de aproximación es 1,092.

Para el problema $P||\sum C_j$, Conway, Maxwell y Miller [22] plantean que la regla SPT permite obtener una solución óptima.

Teorema 2.3 *(Conway, Maxwell y Miller [22]).*
Para el problema $P||\sum C_j$, asignar secuencialmente las tareas de acuerdo a la regla SPT a la máquina con menor carga de trabajo, permite obtener una solución óptima.

2.1.2. Máquinas paralelas no-idénticas

En Cole et al. [21] se estudia una versión descentralizada del problema $R||\sum w_j C_j$, donde se considera que las tareas son administradas por agentes que toman la decisión de dónde asignar su tarea. Esta decisión es tomada considerando su propio beneficio, sin considerar el objetivo global (óptimo social). Para evaluar el desempeño del juego se utiliza el precio de la anarquía, que es el cociente entre el costo de una solución del juego (equilibrio de Nash) y el óptimo del problema centralizado (costo social).

En el trabajo de Cole et al. [21] se define el mapeo $\varphi : \mathcal{M}^{\mathcal{J}} \rightarrow L_2\left([0,\infty)\right)^{\mathcal{M}}$, el cual

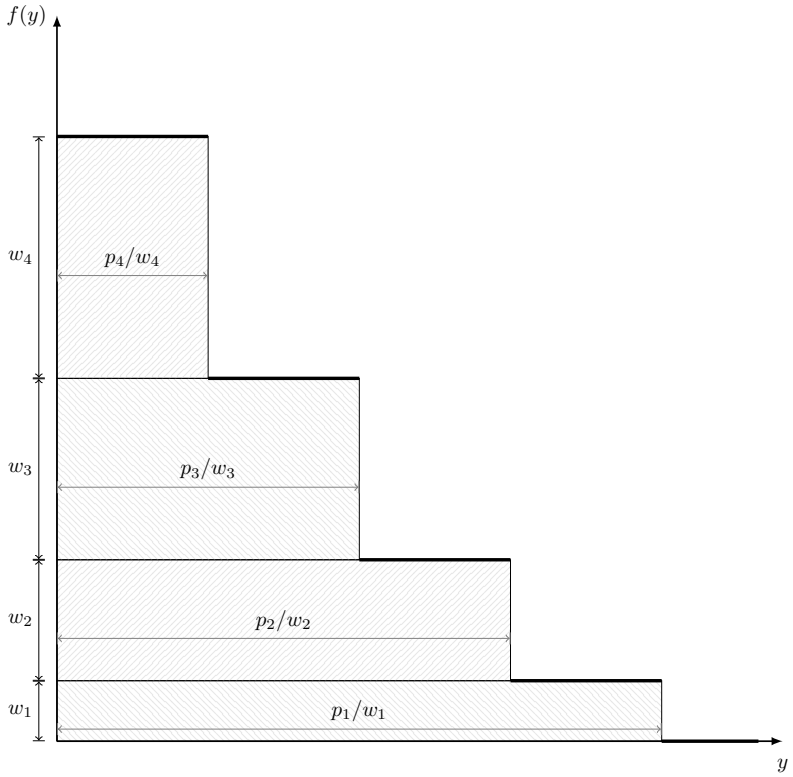


Figura 2.1: Ilustración de la transformada.

fue usado para determinar el precio de anarquía. Este mapeo consiste en asociar, para cada solución factible $\boldsymbol{z}$, un vector de funciones $\boldsymbol{f} = \varphi(\boldsymbol{z})$ (uno para cada máquina), tal que:

$$f_i(y) = \sum_{j \in N_i(\boldsymbol{z}): \frac{p_{ij}}{w_j} \geq y} w_j \tag{2.2}$$

En la figura 2.1 se presenta un ejemplo de la función $f_i(y)$ para una máquina que tiene asignada cuatro tareas. Cada tarea es representada por un rectángulo de área p_j, y se puede observar que el área bajo la función $f_i(y)$ corresponde a la suma de los tiempos de proceso de las tareas asignadas a la máquina.

Sea $\langle f, g \rangle := \int_0^\infty f(y)g(y)dy$ el producto interno usual en L_2. Adicionalmente, se define $\langle \mathbf{f}, \mathbf{g} \rangle := \sum_{i \in \mathcal{M}} \langle f_i, g_i \rangle$, y $||\cdot||$ representa la norma inducida. A continuación se presentan algunas propiedades del mapeo φ. Donde $\boldsymbol{z}$ y $\boldsymbol{z}^*$ representan dos soluciones del problema, tal que $\boldsymbol{f} = \varphi(\boldsymbol{z})$ y $\boldsymbol{f}^* = \varphi(\boldsymbol{z}^*)$.

Teorema 2.4 *(Cole et al. [21]). Propiedades del mapeo φ.*

$$\sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)dy = \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})} p_{ij} \tag{2.3}$$

$$\|\varphi(\boldsymbol{z})\|^2 = \sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)^2 dy = 2C(\boldsymbol{z}) - \eta(\boldsymbol{z}) \tag{2.4}$$

$$\begin{aligned}\langle\varphi(\boldsymbol{z}),\varphi(\boldsymbol{z}^*)\rangle &= \sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)f_i^*(y)\,dy \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{k\in N_i(\boldsymbol{z}^*)} w_j w_k \,\text{mín}\left\{\frac{p_{ik}}{w_k},\frac{p_{ij}}{w_j}\right\}\end{aligned} \tag{2.5}$$

Demostración. Para (2.3).

$$\begin{aligned}\sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)dy &= \sum_{i\in\mathcal{M}}\int_0^\infty \sum_{j\in N_i(\boldsymbol{z}):\ y\le\frac{p_{ij}}{w_j}} w_j dy \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})} w_j\int_0^\infty \mathbb{1}_{y\le\frac{p_{ij}}{w_j}}\,dy \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})} w_j\frac{p_{ij}}{w_j} \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})} p_{ij}\end{aligned}$$

Para (2.5).

$$\begin{aligned}\langle\varphi(\boldsymbol{z}),\varphi(\boldsymbol{z}^*)\rangle &= \sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)f_i^*(y)\,dy \\ &= \sum_{i\in\mathcal{M}}\int_0^\infty \sum_{j\in N_i(\boldsymbol{z}):\ y\le\frac{p_{ij}}{w_j}} w_j \sum_{k\in N_i(\boldsymbol{z}^*):\ y\le\frac{p_{ik}}{w_k}} w_k dy \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{k\in N_i(\boldsymbol{z}^*)} w_j w_k\int_0^\infty \mathbb{1}_{y\le\frac{p_{ij}}{w_j}}\mathbb{1}_{y\le\frac{p_{ik}}{w_k}}\,dy \\ &= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{k\in N_i(\boldsymbol{z}^*)} w_j w_k \,\text{mín}\left\{\frac{p_{ik}}{w_k},\frac{p_{ij}}{w_j}\right\}\end{aligned}$$

Para demostrar (2.4), se usarán las expresiones (1.1), (1.2) y (2.5):

$$\begin{aligned}
\|\varphi(\boldsymbol{z})\|^2 &= \sum_{i\in\mathcal{M}}\int_0^\infty f_i(y)^2\,dy \\
&= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{k\in N_i(\boldsymbol{z})} w_j w_k \,\text{mín}\left\{\frac{p_{ik}}{w_k},\frac{p_{ij}}{w_j}\right\} \\
&= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{k\in N_i(\boldsymbol{z})}\left(w_j p_{ik}\mathbb{1}_{\frac{p_{ik}}{w_k}\leq\frac{p_{ij}}{w_j}} + w_k p_{ij}\mathbb{1}_{\frac{p_{ik}}{w_k}\not\leq\frac{p_{ij}}{w_j}}\right) \\
&= \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{\substack{k\in N_i(\boldsymbol{z})\\ k\preceq j}} w_j p_{ik} + \sum_{i\in\mathcal{M}}\sum_{j\in N_i(\boldsymbol{z})}\sum_{\substack{k\in N_i(\boldsymbol{z})\\ k\succ j}} w_k p_{ij} \\
&= 2C(\boldsymbol{z}) - \eta(\boldsymbol{z})
\end{aligned}$$

De manera similar se puede demostrar que $\|\varphi(\boldsymbol{z}^*)\|^2 = 2C(\boldsymbol{z}^*) - \eta(\boldsymbol{z}^*)$. □

En Cole et al. [21] se estudian mecanismos descentralizados para juegos en el problema $R||\sum w_jC_j$. Determinan que el precio de la anarquía cuando se utiliza la regla de Smith es a lo más 4. Para el caso de donde las tareas son interrumpibles se determina que el precio de la anarquía es 2,618 para el caso ponderado, y 2,5 para el caso no ponderado.

En Caragiannis et al. [18] se estudia el problema de balanceo de carga, en donde un conjunto de clientes debe asignar una tarea a algún servidor de un conjunto restringido de servidores. Presentan resultados para ambientes de máquinas restringidas idénticas (RP) y no idénticas (RQ y R) para los casos ponderados y no ponderados. Estos resultados permiten determinar la calidad de las soluciones obtenidas de manera descentralizada y aleatoria, y son obtenidos por medio de análisis de peor caso.

En Abed [2] y Abed et al.[3], se estudian mecanismos de coordinación para juegos de programación de la producción en el problema $R||\sum w_jC_j$. Entre sus resultados, determinaron que para el problema centralizado se tiene un factor de aproximación de 2,618 para las soluciones jump-óptimas y swap-óptimas, incluso para el caso donde un agente controla todas las tareas.

2.2. Reducción de elegibilidad

La reducción de elegibilidad de máquina, será utilizada para determinar cotas superiores para el problema en los ambientes R, RQ y RP.

Definición 2.5 ***Reducción de elegibilidad de máquina.***
Sea $\mathcal{I}$ una instancia del problema $R||\sum w_jC_j$, donde la tarea j puede ser procesada por un conjunto de máquinas $\mathcal{M}_j \subseteq \mathcal{M}$. Esto es, $p_{ij} = \infty$, para todo $i \notin \mathcal{M}_j$. Sea $\boldsymbol{x}^*$ una solución óptima de $\mathcal{I}$, donde la tarea j es procesada por la máquina x_j^*. Además, sea $\boldsymbol{x}$ alguna solución jump-óptima, donde la tarea j es procesada por la máquina x_j. La reducción de elegibilidad de máquina consiste en reducir la cantidad de máquinas que pueden procesar cada tarea $j \in \mathcal{J}$.

Esto es, $\mathcal{M}'_j = \{x_j, x_j^*\} \subseteq \mathcal{M}_j$ *el conjunto reducido de máquinas que pueden procesar la tarea* j*. En particular, se tendrá que* $|\mathcal{M}'_j| = 1$ *si* $x_j = x_j^*$*.*

Lema 2.6 *Sea* $\mathcal{I}$ *una instancia del problema* $R||\sum w_j C_j$*,* $\mathcal{I}'$ *la instancia que se obtiene al aplicar la reducción de elegibilidad de máquina a* $\mathcal{I}$*. Dados* $\boldsymbol{x}$ *y* $\boldsymbol{x}^*$*, solución jump-óptima y óptima global respectivamente. El factor de aproximación para* $\boldsymbol{x}$ *es el mismo en ambas instancias,* $\mathcal{I}$ *y* $\mathcal{I}'$*.*

Demostración. Por contradicción, supondremos que $\boldsymbol{x}$ no es un jump-óptimo de la instancia $\mathcal{I}'$. Dado que $\boldsymbol{x}$ no es un jump-óptimo de $\mathcal{I}'$, es posible mejorar $C(\boldsymbol{x})$ realizando una movida de jump para alguna tarea $k \in \mathcal{J}$. Para esta tarea se tiene $\mathcal{M}'_k = \{x_k, x_k^*\}$. Luego, la tarea k se mueve a la máquina x_k^*, reduciendo el costo a $C(\tilde{\boldsymbol{x}}) < C(\boldsymbol{x})$. Pero $\mathcal{M}'_k \subseteq \mathcal{M}_k$, luego $\boldsymbol{x}$ no sería un jump-óptimo de $\mathcal{I}$. Por otra parte, $\boldsymbol{x}^*$ es el óptimo global de $\mathcal{I}$, por construcción también es factible en $\mathcal{I}'$. Como toda solución de $\mathcal{I}'$ es también solución de $\mathcal{I}$, $\boldsymbol{x}^*$ es un óptimo global de $\mathcal{I}'$. Luego, el factor de aproximación es el mismo para ambas instancias, $\mathcal{I}$ y $\mathcal{I}'$. □

El resultado anterior, se puede generalizar a cualquier ambiente de máquinas paralelas. Dado que los ambientes de máquinas idénticas (P), idénticas restringidas (RP), uniformes (Q) y uniformes restringidas (RQ), son casos particulares del ambiente de máquinas paralelas no-relacionadas (R).

2.3. Desigualdades

En esta sección se presentan desigualdades que serán utilizadas para determinar cotas superiores para el problema en los ambientes R, RQ y RP. Las desigualdades presentadas en los Lemas 2.7 y 2.8 son resultado del trabajo realizado en esta tesis. Mientras que la desigualdad del Lema 2.9 es trivial y ampliamente usada en la literatura.

Lema 2.7 *Para cada par de números enteros no negativos* a *y* b*, se tiene:*

$$ab \leq \frac{a^2}{2} + \frac{b^2}{2} - \frac{a}{2} + \frac{b}{2}$$

Demostración. La desigualdad puede ser escrita como $f(a,b) = \left(\frac{a-b}{2}\right)(a-b-1) \geq 0$.
Si $a-b > 0 \Rightarrow a > b$, como a y b son enteros no negativos, se tiene que $a \geq b+1 \Rightarrow a-b-1 \geq 0$.
Luego, $f(a,b) \geq 0,\ \forall a > b$.
Si $a - b < 0 \Rightarrow a < b \Rightarrow a < b+1 \Rightarrow a - b - 1 < 0$. Luego, $f(a,b) > 0,\ \forall a < b$.
Finalmente, si $a = b$ se tiene que $f(a,b) = 0$.
Por lo tanto, $f(a,b) \geq 0$ cuando a y b son enteros no negativos. □

Lema 2.8 *Para cada par de números enteros no negativos* a *y* b*, se tiene:*

$ab \leq \left(1 - \frac{\phi}{2}\right)a^2 + \left(1 + \frac{1}{2\phi}\right)b^2 + \frac{\phi}{2}a - \left(1 + \frac{1}{2\phi}\right)b$*; donde* $\phi = \frac{1+\sqrt{5}}{2}$ *es el número áureo*

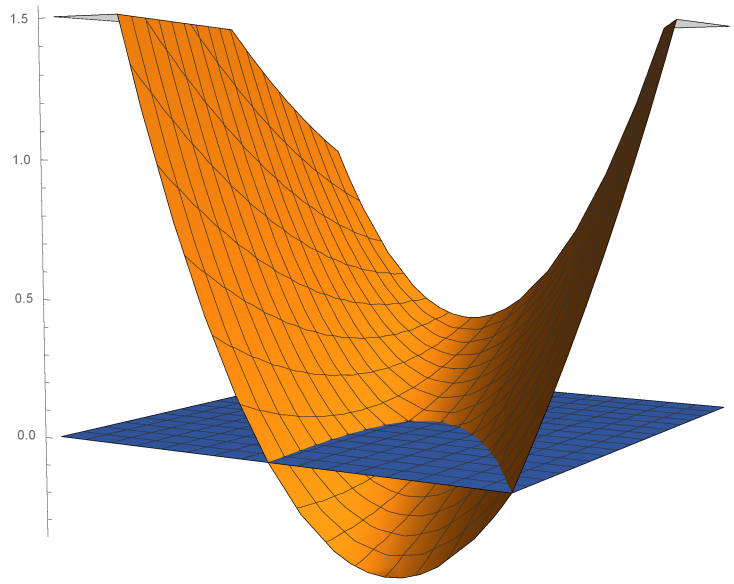
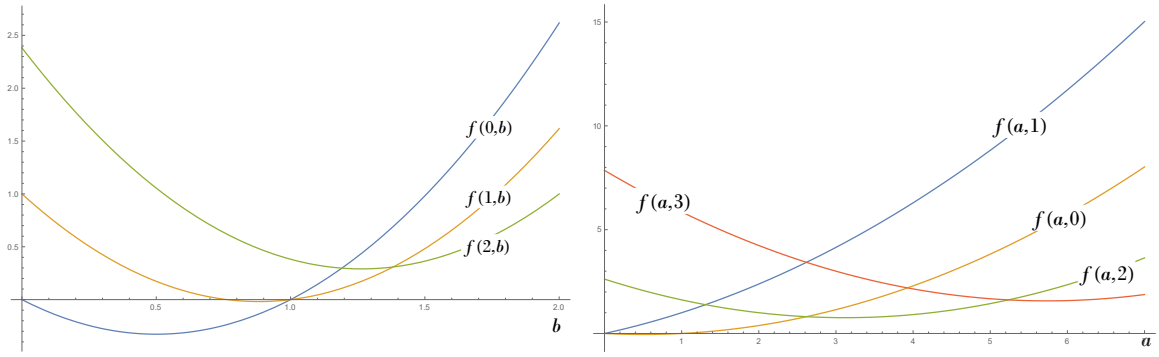


Figura 2.2: Superficie de la desigualdad.



(a) a fijo. (b) b fijo.

Figura 2.3: Curvas de nivel para $f(a,b)$

Demostración. La desigualdad puede ser escrita como:
$f(a,b) = \left(1-\frac{\phi}{2}\right)a^2 + \left(1+\frac{1}{2\phi}\right)b^2 + \frac{\phi}{2}a - \left(1+\frac{1}{2\phi}\right)b - ab \geq 0$. En la figura 2.2 se muestra la superficie de $f(a,b)$, mientras que en la figura 2.3 se presentan algunas curvas de nivel.
Para valores particulares de a, y b se tienen los siguientes resultados:
$f(0,0) = f(1,1) = 0$.
$f(a,0) = \left(1-\frac{\phi}{2}\right)a^2 + \frac{\phi}{2}a = \frac{a}{2}\left((2-\phi)a+\phi\right) \geq 0$, para todo $a \geq 0$.
$f(0,b) = \left(1+\frac{1}{2\phi}\right)b^2 - \left(1+\frac{1}{2\phi}\right)b = \left(1+\frac{1}{2\phi}\right)b(b-1) \geq 0$, para todo $b \geq 1$.
Dado lo anterior, solo falta demostrar que $f(a,b) \geq 0$, para $a, b \geq 1$.

Sea el siguiente modelo de programación no-lineal:

$$\begin{aligned} \text{[D]: maximizar} \quad & f(a,b) \\ \text{sujeto a} \quad & a \geq 1 \qquad (2.6) \\ & b \geq 1 \qquad (2.7) \end{aligned}$$

La función $f(a,b)$ es una función cuadrática, cuya forma cuadrática tiene valores propios $\gamma = \{0, \frac{3}{2}\}$. Entonces, esta forma cuadrática es semidefinida positiva y $f(a,b)$ es una función convexa.
Para el problema [D], como $f(a,b)$ es convexa, y las restricciones (2.6) y (2.7) son lineales, la solución encontrada al resolver las condiciones de Karush-Kuhn-Tucker (KKT), será un óptimo global restringido.
Usando λ_a y λ_b como los multiplicadores de Lagrange de (2.6) y (2.7) respectivamente. Se plantean las condiciones de KKT para el problema [D]:

$$\begin{aligned} (2-\phi)a - b - \lambda_a + \frac{\phi}{2} &= 0 \\ -a + \left(2 - \frac{1}{\phi}\right) b - \lambda_b - \left(1 + \frac{1}{2\phi}\right) &= 0 \\ \lambda_a(a-1) &= 0 \\ \lambda_b(b-1) &= 0 \\ a, b &\geq 1 \\ \lambda_a, \lambda_b &\geq 0 \end{aligned}$$

La solución para las condiciones de KKT se obtiene cuando las restricciones (2.6) y (2.7) son activas. Donde: $a^* = b^* = 1$; $\lambda_a^* = 1 - \frac{\phi}{2}$; $\lambda_b^* = \frac{1}{2\phi}$; $f(a^*, b^*) = 0$.
Por lo tanto, $f\,(a,b) \geq 0$, cuando a y b son enteros no negativos. □

Lema 2.9 *Para cada par de números reales no negativos a y b, se tiene:*

$$ab \leq \frac{\beta}{2}a^2 + \frac{1}{2\beta}b^2, \ \textit{donde } \beta \in \mathbb{R}$$

Demostración. Directamente de $\left(\beta^{\frac{1}{2}}a - \beta^{-\frac{1}{2}}b\right)^2 \geq 0$. □

Capítulo 3

Máquinas paralelas idénticas

En este capítulo se presentan resultados que permiten determinar la calidad de las soluciones jump-óptimas para el problema de minimizar el tiempo ponderado de completación en máquinas paralelas idénticas $P||\sum w_jC_j$. Se presentan algunas condiciones necesarias para la existencia de un problema de aproximación por búsqueda local. Estas condiciones permiten determinar si una instancia es trivial para ser resuelta por búsqueda local usando un vecindario de jump. Además del caso general, se consideran los casos: no ponderado $P||\sum C_j$ y tiempo de proceso constante ponderado $P|p_j = 1|\sum w_jC_j$.

Para todas las variantes del problema, se determinó que el factor de aproximación de las soluciones jump-óptimas no es mayor que $\frac{3}{2} - \frac{1}{2m}$. Para el problema $P||\sum w_jC_j$ se tiene una cota inferior de $\frac{6}{5}$, mientras que para los problemas $P||\sum C_j$ y $P|p_j = 1|\sum w_jC_j$ esta cota es de $\frac{15}{13}$.

También se presentan factores de aproximación no-constantes para instancias de los problemas $P||\sum w_jC_j$ y $P||\sum C_j$.

3.1. Introducción

Para describir el problema y sus variantes se define la siguiente notación:

$\mathcal{J} = \{1, \ldots, n\}$: el conjunto tareas;

$\mathcal{M} = \{1, \ldots, m\}$: el conjunto de máquinas;

p_j: tiempo necesario para procesar la tarea $j \in \mathcal{J}$;

w_j: peso o importancia relativa de la tarea $j \in \mathcal{J}$;

$\boldsymbol{z}$: una solución factible del problema;

$C_j(\boldsymbol{z})$: tiempo de completación de la tarea j en la solución $\boldsymbol{z}$;

$C(\boldsymbol{z}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{z})$: costo de la solución $\boldsymbol{z}$;

$N_i(\boldsymbol{z})$: conjunto de tareas asignadas a la máquina i en la solución $\boldsymbol{z}$;

En este ambiente de máquinas, cada tarea $j \in \mathcal{J}$ puede ser procesada, sin interrupción, por cualquier máquina $i \in \mathcal{M}$, requiriendo un tiempo de proceso p_j. Cada máquina puede procesar una tarea a la vez, y todas las tareas y máquinas están disponibles desde el inicio. El problema consiste en determinar una solución $\boldsymbol{z}$, que minimice $C(\boldsymbol{z})$. La solución $\boldsymbol{z}$, establece la máquina a la que cada tarea es asignada y la secuencia en que las tareas son procesadas. Para cada máquina, la secuencia es definida por las reglas WSPT y SPT, para los objetivos $\sum w_j C_j$ y $\sum C_j$ respectivamente (Teoremas 1.1 y 1.2).

Una instancia $\mathcal{I}$ del problema, queda completamente definida por $\mathcal{J}$, $\mathcal{M}$, $\boldsymbol{w}$ y $\boldsymbol{p}$. Donde $\boldsymbol{w}$ y $\boldsymbol{p}$ representan el vector de pesos y tiempos de proceso de las tareas respectivamente.

Sea $\boldsymbol{z}$ cualquier solución factible de la instancia $\mathcal{I}$. El costo de la solución $\boldsymbol{z}$ puede ser determinado por (3.1), donde el símbolo $k \prec j$ indica que la tarea k precede a la tarea j. Por otra parte $k \succ j$ indica que la tarea k sucede a la tarea j (los símbolos $\preceq$ y $\succeq$ además incluyen la tarea j). La precedencia entre las tareas se define por la regla WSPT o SPT según corresponda. En los casos donde exista empate entre dos o más tareas, su precedencia se define en forma arbitraria.

$$C(\boldsymbol{z}) = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\prec j}} w_j p_k = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succ j}} w_k p_j \tag{3.1}$$

Para cualquier instancia $\mathcal{I}$, la asignación óptima de tareas a las máquinas es representada por el vector $\boldsymbol{x}^*$. Donde x_j^* indica la máquina donde es asignada la tarea j, es decir, $x_j^* = i$ si la tarea j es programada en la máquina i. Adicionalmente se define:

$N_i(\boldsymbol{x}^*) = \left\{j \in \mathcal{J} : x_j^* = i\right\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}^*$;

$C_j(\boldsymbol{x}^*)$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}^*$;

$C(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x}^*)$: el costo óptimo.

La solución jump-óptima es representada por el vector $\boldsymbol{x}$. Donde x_j indica la máquina donde es asignada la tarea j, es decir, $x_j = i$ si la tarea j es programada en la máquina i en el óptimo local. Adicionalmente se define:

$N_i(\boldsymbol{x}) = \{j \in \mathcal{J} : x_j = i\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}$;

$C_j(\boldsymbol{x})$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}$;

$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x})$: el costo de la solución jump-óptima $\boldsymbol{x}$.

Con lo anterior y (3.1), es posible determinar los costos de las soluciones óptima y jump-óptima como:

$$C(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\prec j}} w_j p_k = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\succ j}} w_k p_j \tag{3.2}$$

$$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\prec j}} w_j p_k = \sum_{j\in\mathcal{J}} w_j p_j + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k p_j \tag{3.3}$$

En notación de scheduling el problema es representado por $P||\sum w_j C_j$. A partir de este problema general, es posible obtener algunos casos particulares (las equivalencias quedan demostradas por el Lema 1.3):

1. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son arbitrarios, se tiene el problema $P||\sum w_j C_j$
2. Si $\boldsymbol{w}$ es constante, el problema resultante es equivalente a $P||\sum C_j$
3. Si $\boldsymbol{p}$ es constante, el problema resultante es equivalente a $P|p_j = 1|\sum w_j C_j$
4. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son constantes, el problema resultante es equivalente a $P|p_j = 1|\sum C_j$. En el Lema 3.2 se demuestra que este problema es trivial, dado que el costo de cualquier solución óptima es idéntica a la de cualquier solución jump-óptima.

En las siguientes secciones se estudia la calidad de las soluciones jump-óptimas para el problema $P||\sum w_j C_j$ y su variantes, considerando factores de aproximación constantes y no-constantes. Para las instancias de cada variante de $P||\sum w_j C_j$, se determinó un factor de aproximación no-constante ajustado. Para determinar el factor constante de aproximación de las soluciones jump-óptimas, se establecen cotas superiores e inferiores. Para establecer las cotas superiores se utilizaron los resultados presentados por Brueggemann et al. [15] y Eastman et al. [27]. Para las cotas inferiores se presentan instancias que exponen un peor caso para los problemas.

En la Sección 3.2 se presentan las condiciones necesarias para la existencia de un problema de aproximación por búsqueda local para el vecindario de jump, las que permiten determinar si una instancia es trivial para ser resuelta por búsqueda local. Esto es, que el costo de cualquier solución jump-óptima sea distinto al costo de la solución óptima.

En las Secciones 3.3, 3.4 y 3.5 se presentan resultados para los problemas $P||\sum w_j C_j$, $P||\sum C_j$ y $P|p_j = 1|\sum w_j C_j$ respectivamente.

3.2. Condiciones necesarias para la existencia de un problema de aproximación

En esta sección se presentan dos condiciones necesarias para que exista un problema de aproximación en el problema $P||\sum w_j C_j$. Estas, establecen las condiciones que debe cumplir

cualquier instancia de $P||\sum w_jC_j$, para que el costo de cualquier solución jump-óptima sea distinto al costo del óptimo global.

Lema 3.1 *Para el problema $P||\sum w_jC_j$, si $n \leq m+1$ el costo de cualquier jump-óptimo es el mismo del óptimo global.*

Demostración. Para los casos donde $n \leq m$, resulta trivial que $C(\boldsymbol{x}) = C(\boldsymbol{x}^*)$, dado que en las soluciones $\boldsymbol{x}^*$ y $\boldsymbol{x}$ se asignará a lo más una tarea a cada máquina. Para los casos donde $n = m+1$, esta conclusión no resulta trivial. Por contradicción, se demostrará que para este caso $C(\boldsymbol{x}) = C(\boldsymbol{x}^*)$.

Sea $\mathcal{I}$ una instancia con m máquinas idénticas y $n = m+1$ tareas, tal que $C(\boldsymbol{x}^*) < C(\boldsymbol{x})$. Dado que hay $n = m+1$ tareas, en cualquier solución jump-óptima se tendrá que todas las máquinas, excepto una, procesarán una tarea. En la solución óptima $\boldsymbol{x}^*$ las tareas j_1 y j_2 son asignadas a la misma máquina, y las tareas j_α y j_β son asignadas a máquinas distintas. Mientras que en la solución jump-óptima $\boldsymbol{x}$, las tareas j_1 y j_2 son asignadas en máquinas distintas, y las tareas j_α y j_β son asignadas a la misma máquina.

Sin pérdida de generalidad, se considera que $j_\alpha \prec j_\beta$ y $j_1 \prec j_2$. Luego, el costo de $\boldsymbol{x}^*$ es $C(\boldsymbol{x}^*) = \eta + w_2p_1$, mientras que el costo de $\boldsymbol{x}$ es $C(\boldsymbol{x}) = \eta + w_\beta p_\alpha$. Donde $\eta = \sum_{j\in\mathcal{J}} w_jp_j$.

Entonces, como $C(\boldsymbol{x}^*) < C(\boldsymbol{x})$, se tiene que:

$$w_2p_1 < w_\beta p_\alpha \tag{3.4}$$

En la solución jump-óptima[1] $\boldsymbol{x}$, la máquina 1 procesa las tareas j_α y j_β, la máquina i procesa la tarea j_{i-1}, para $i = 2, \ldots, m$. En la tabla 3.1 se presentan los detalles de la solución $\boldsymbol{x}$, y las soluciones que se pueden obtener por movidas de jump que ofrecen potencial para reducir el costo.

Solución	$N_1(\cdot)$	$N_2(\cdot)$	$N_3(\cdot)$	$N_i(\cdot)$, $4 \leq i \leq m$	Costo
$\boldsymbol{x}$	$\{j_\alpha, j_\beta\}$	$\{j_1\}$	$\{j_2\}$	$\{j_{i-1}\}$	$\eta + w_\beta p_\alpha$
$\boldsymbol{x}_1$	$\{j_\alpha\}$	$\{j_1, j_\beta\}$	$\{j_2\}$	$\{j_{i-1}\}$	$\eta + w_\beta p_1 \mathbb{1}_{j_1 \prec j_\beta} + w_1 p_\beta \mathbb{1}_{j_1 \succ j_\beta}$
$\boldsymbol{x}_2$	$\{j_\alpha\}$	$\{j_1\}$	$\{j_2, j_\beta\}$	$\{j_{i-1}\}$	$\eta + w_\beta p_2 \mathbb{1}_{j_2 \prec j_\beta} + w_2 p_\beta \mathbb{1}_{j_2 \succ j_\beta}$
$\boldsymbol{x}_3$	$\{j_\beta\}$	$\{j_1, j_\alpha\}$	$\{j_2\}$	$\{j_{i-1}\}$	$\eta + w_\alpha p_1 \mathbb{1}_{j_1 \prec j_\alpha} + w_1 p_\alpha \mathbb{1}_{j_1 \succ j_\alpha}$
$\boldsymbol{x}_4$	$\{j_\beta\}$	$\{j_1\}$	$\{j_2, j_\alpha\}$	$\{j_{i-1}\}$	$\eta + w_\alpha p_2 \mathbb{1}_{j_2 \prec j_\alpha} + w_2 p_\alpha \mathbb{1}_{j_2 \succ j_\alpha}$

Tabla 3.1: Soluciones con potencial para reducir el costo.

Para que la solución $\boldsymbol{x}$ sea un jump-óptimo se requiere que se cumplan las siguientes condi-

[1] Existe simetría en las soluciones por tratarse de máquinas paralelas idénticas.

ciones de optimalidad local

$$\begin{aligned}
C(\boldsymbol{x}) \leq C(\boldsymbol{x}_1) &\Rightarrow w_\beta p_\alpha \leq w_\beta p_1 \mathbb{1}_{j_1 \prec j_\beta} + w_1 p_\beta \mathbb{1}_{j_1 \succ j_\beta} \\
C(\boldsymbol{x}) \leq C(\boldsymbol{x}_2) &\Rightarrow w_\beta p_\alpha \leq w_\beta p_2 \mathbb{1}_{j_2 \prec j_\beta} + w_2 p_\beta \mathbb{1}_{j_2 \succ j_\beta} \\
C(\boldsymbol{x}) \leq C(\boldsymbol{x}_3) &\Rightarrow w_\beta p_\alpha \leq w_\alpha p_1 \mathbb{1}_{j_1 \prec j_\alpha} + w_1 p_\alpha \mathbb{1}_{j_1 \succ j_\alpha} \\
C(\boldsymbol{x}) \leq C(\boldsymbol{x}_4) &\Rightarrow w_\beta p_\alpha \leq w_\alpha p_2 \mathbb{1}_{j_2 \prec j_\alpha} + w_2 p_\alpha \mathbb{1}_{j_2 \succ j_\alpha}
\end{aligned} \tag{3.5}$$

Para determinar si la solución $\boldsymbol{x}$ es un jump-óptimo, es necesario definir y analizar todas las posibilidades de acuerdo a la precedencia de las tareas. Dado que se asumió que $j_\alpha \prec j_\beta$ y $j_1 \prec j_2$, se tienen las siguientes precedencias posibles entre las tareas j_α, j_β, j_1 y j_2:

(a) $j_\alpha \prec j_1 \prec j_2 \prec j_\beta$

(b) $j_\alpha \prec j_1 \prec j_\beta \prec j_2$

(c) $j_\alpha \prec j_\beta \prec j_1 \prec j_2$

(d) $j_1 \prec j_\alpha \prec j_\beta \prec j_2$

(e) $j_1 \prec j_\alpha \prec j_2 \prec j_\beta$

(f) $j_1 \prec j_2 \prec j_\alpha \prec j_\beta$

Caso (a). De (3.5), se tiene que $p_\alpha \leq \min\{p_1, p_2\}$ y $w_\beta \leq \min\{w_1, w_2\}$. Luego, por (3.4) se tiene la siguiente contradicción $w_2 p_1 < \min\{w_1, w_2\} \min\{p_1, p_2\}$.

Caso (b). De (3.5), se tiene que $p_\alpha \leq p_1$ y $w_\beta \leq \min\{w_1, w_2\}$. Luego, por (3.4) se tiene la siguiente contradicción $w_2 p_1 < p_1 \min\{w_1, w_2\}$.

Caso (c). De (3.5), se tiene que $p_\alpha \leq \frac{p_\beta}{w_\beta} \min\{w_1, w_2\}$ y $w_\beta \leq \min\{w_1, w_2\}$. Adicionalmente, para este caso se tiene $\frac{p_\beta}{w_\beta} \leq \frac{p_1}{w_1}$. Luego, por (3.4) se tiene la siguiente contradicción $w_1 w_2 < (\min\{w_1, w_2\})^2$.

Caso (d). De (3.5), se tiene que $p_\alpha \leq p_1$ y $w_\beta \leq w_2$. Luego, por (3.4) se tiene la siguiente contradicción $w_2 p_1 < w_2 p_1$.

Caso (e). De (3.5), se tiene que $p_\alpha \leq \min\{p_1, p_2\}$ y $w_\beta \leq w_2$. Luego, por (3.4) se tiene la siguiente contradicción $w_2 p_1 < w_2 \min\{p_1, p_2\}$.

Caso (f). De (3.5), se tiene que $p_\alpha \leq \min\{p_1, p_2\}$ y $w_\beta \leq \frac{w_\alpha}{p_\alpha} \min\{p_1, p_2\}$. Adicionalmente, para este caso se tiene $\frac{w_\alpha}{p_\alpha} \leq \frac{w_2}{p_2}$. Luego, por (3.4) se tiene la siguiente contradicción $p_1 p_2 < (\min\{p_1, p_2\})^2$.

Como todas las precedencias posibles contradicen (3.4), se tiene que $C(\boldsymbol{x}^*) \not< C(\boldsymbol{x})$. Por lo tanto, para cualquier instancia $\mathcal{I}$ con $n \leq m+1$ el costo de cualquier solución jump-óptima $\boldsymbol{x}$ será el mismo que el costo de la solución óptima $\boldsymbol{x}^*$. □

En el Lema 3.3, se presenta una instancia del problema $P||\sum w_j C_j$ con $n = m+2$, donde $C(\boldsymbol{x}) > C(\boldsymbol{x}^*)$.

El problema $P|p_j = 1|\sum C_j$ es un caso particular del problema $P||\sum w_j C_j$, donde $\boldsymbol{w}$ y $\boldsymbol{p}$ son vectores unitarios. Por el Lema 1.3, se tiene que este problema es equivalente al caso

donde los pesos y tiempos de proceso de las tareas son constantes. Esto es, $w_j = w$ y $p_j = p$ para todo $j \in \mathcal{J}$.

Lema 3.2 *Cualquier solución jump-óptima del problema* $P|p_j = 1|\sum C_j$ *tiene el mismo costo del óptimo global.*

Demostración. Sea $\mathcal{I}$ una instancia con m máquinas y n tareas, cuyo tiempo de proceso es unitario. Sin pérdida de generalidad, se considera que las máquinas están ordenadas en orden decreciente de acuerdo a la cantidad de tareas que tienen asignadas en la solución jump-óptima $\boldsymbol{x}$. Esto es, $|N_1(\boldsymbol{x})| \geq |N_2(\boldsymbol{x})| \geq \cdots \geq |N_m(\boldsymbol{x})|$.
Luego, al retirar cualquier tarea de la primera máquina se consigue la mayor reducción en el costo $C(\boldsymbol{x})$. Sea $j \in N_1(\boldsymbol{x})$ la tarea movida y $\delta = |N_1(\boldsymbol{x})|$ la reducción en el costo $\boldsymbol{x}$. Por otra parte, el menor aumento en el costo se consigue insertando la tarea j en la máquina m, este aumento en el costo es $\delta' = 1 + |N_m(\boldsymbol{x})|$. Luego, por condición de optimalidad local se debe cumplir $\delta \leq \delta'$.
Entonces, $|N_1(\boldsymbol{x})| - |N_m(\boldsymbol{x})| \leq 1$. Por lo tanto, la máquina 1 puede tener asignado a lo más una tarea adicional con respecto a la cantidad de tareas que tiene la máquina m.
Sea k la parte entera de n/m, y r el resto, tal que $km + r = n$. Luego, en $\boldsymbol{x}$, se tiene que r máquinas tendrán asignadas $k+1$ tareas, y las restantes máquinas tendrán k tareas. Dado que las tareas tienen tiempo de proceso unitario, el aporte al costo total de una máquina que procesa q tareas es $\frac{q(q+1)}{2}$. Por lo tanto, el costo del jump-óptimo $\boldsymbol{x}$ es:

$$C(\boldsymbol{x}) = \frac{r(k+1)(k+2)}{2} + \frac{(m-r)k(k+1)}{2} = \frac{(k+1)(r+n)}{2}$$

Por otra parte, para construir la solución óptima $\boldsymbol{x}^*$, se utiliza el Teorema 2.3. En la solución óptima $\boldsymbol{x}^*$, todas las máquinas tendrán k tareas, generando un costo $\frac{mk(k+1)}{2}$. Adicionalmente se tendrá que r máquinas tendrán una tarea adicional, generando un aporte de $r(k+1)$. Luego, el costo total será:

$$C(\boldsymbol{x}^*) = \frac{mk(k+1)}{2} + r(k+1) = \frac{(k+1)(r+n)}{2}$$

Finalmente, como $C(\boldsymbol{x}^*) = C(\boldsymbol{x})$ se completa la demostración. □

3.3. Tiempo ponderado de completación

En la notación de scheduling este problema se representa por $P||\sum w_j C_j$. Complementando la notación definida en la Sección 3.1, se define $\tilde{C}$ como el tiempo ponderado de completación si todas las tareas son asignadas en una máquina.

$$\tilde{C} = \sum_{j \in \mathcal{J}} w_j \sum_{\substack{k \in \mathcal{J} \\ k \prec j}} p_k + \sum_{j \in \mathcal{J}} w_j p_j = \sum_{j \in \mathcal{J}} p_j \sum_{\substack{k \in \mathcal{J} \\ k \succ j}} w_k + \sum_{j \in \mathcal{J}} w_j p_j \tag{3.6}$$

En la figura 3.1 se presenta un esquema de la movida de jump para la tarea j. Al asignar la tarea j a la máquina t, se produce un cambio en su tiempo de completación, este será la

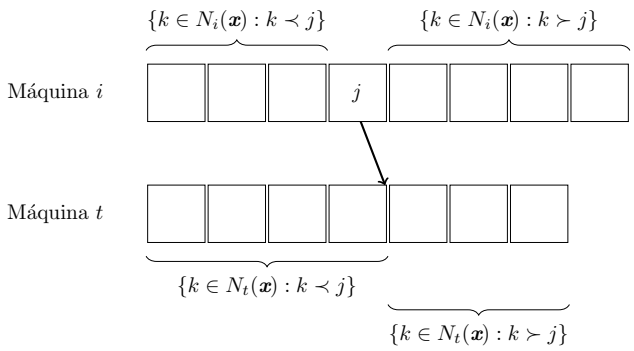


Figura 3.1: Esquema para movida de jump en $P||\sum w_j C_j$

suma entre su tiempo de proceso p_j y el tiempo de proceso de todas las tareas asignadas a la máquina t que preceden a la tarea j. Al retirar la tarea j de la máquina i, el tiempo de completación de todas las tareas que suceden a la tarea j se reduce en p_j. Al asignar la tarea j a la máquina t, el tiempo de completación de todas las tareas que suceden a la tarea j en la máquina t aumenta en p_j.

Para establecer una cota inferior para el factor de aproximación, se presenta una instancia del problema que expone un peor caso.

	j_1	j_2	j_3	j_4
p_j	1	1	2	2
w_j	2	2	1	1

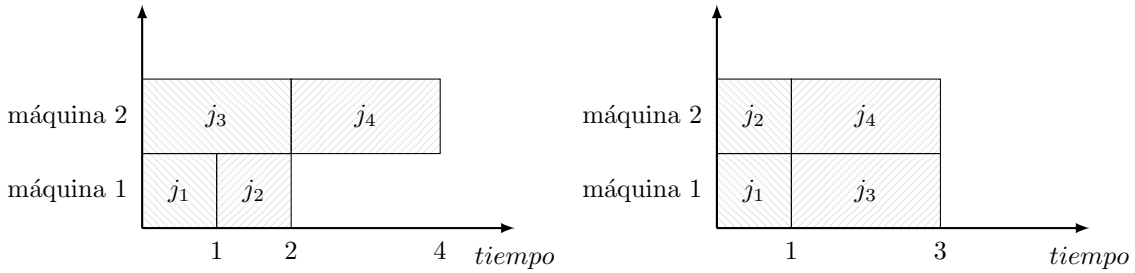
Tabla 3.2: Instancia peor caso $P||\sum w_j C_j$

Lema 3.3 *El factor de aproximación de una solución jump-óptima para el problema* $P||\sum w_j C_j$, *es al menos* $\frac{6}{5}$.

Demostración. Sea una instancia con $m = 2$ máquinas y $n = 4$ tareas. En la tabla 3.2 se presentan los pesos y tiempos de proceso de las tareas. El costo óptimo para esta instancia es $C(\boldsymbol{x}^*) = 10$, mientras que el peor jump-óptimo tiene un costo de $C(\boldsymbol{x}) = 12$. Luego, el factor de aproximación es $\frac{6}{5}$. El peor jump-óptimo se consigue asignando las tareas j_1 y j_2 en alguna de las máquinas, y las tareas j_3 y j_4 en la otra máquina. El óptimo global se consigue asignando las tareas j_1 y j_3 en alguna de las máquinas, y las tareas j_2 y j_4 en la otra máquina. En la figura 3.2 se presentan cartas Gantt de estas soluciones.

□

A continuación se presenta una cota superior para el factor de aproximación, la cual es una demostración alternativa a la propuesta por Brueggemann et al. [15] (ver Teorema 2.2). Para determinar la cota superior del factor de aproximación, se utiliza un resultado de Eastman et al. [27], presentado en Teorema 2.1.



(a) Solución jump-óptima. (b) Solución óptima.

Figura 3.2: Soluciones para instancia de $P||\sum w_jC_j$

Teorema 3.4 *El factor de aproximación de una solución jump-óptima para el problema* $P||\sum w_jC_j$, *se encuentra entre* $\frac{6}{5}$ *y* $\frac{3}{2}-\frac{1}{2m}$.

Demostración. Sea $\boldsymbol{x}^*$ la solución óptima y $\boldsymbol{x}$ una solución jump-óptima; $\delta_{ij}(\boldsymbol{x})$ el ahorro en $C(\boldsymbol{x})$ al retirar la tarea j de la máquina i, y δ'_{tj} el incremento en $C(\boldsymbol{x})$ al asignar la tarea j a la máquina t.

$$\delta_{ij} = w_jC_j(\boldsymbol{x}) + \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} w_kp_j$$

$$\delta'_{tj} = w_jp_j + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j$$

Para que $\boldsymbol{x}$ sea un óptimo local, se debe cumplir que $\delta_{ij} \le \delta'_{tj}$ para todo $j \in N_i(\boldsymbol{x}), \text{con } i \in \mathcal{M}, t \in \mathcal{M}\setminus\{i\}$. Luego, se tiene:

$$w_jC_j(\boldsymbol{x}) + \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} w_kp_j \le w_jp_j + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j \tag{3.7}$$

Sumando para todo $t \in \mathcal{M}\setminus\{i\}$

$$(m-1)w_jC_j(\boldsymbol{x})+(m-1)\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} w_kp_j \le (m-1)w_jp_j+\sum_{t\in\mathcal{M}\setminus\{i\}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k+\sum_{t\in\mathcal{M}\setminus\{i\}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j$$

Considerando las siguientes dos expresiones

$$\sum_{t\in\mathcal{M}\setminus\{i\}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k = \sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k - \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\prec j}} w_jp_k$$

$$\sum_{t\in\mathcal{M}\setminus\{i\}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j = \sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j - \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} w_kp_j$$

Se tiene

$$(m-1)w_jC_j(\boldsymbol{x})+m\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} w_kp_j+\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\prec j}} w_jp_k \le (m-1)w_jp_j+\sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} w_jp_k+\sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} w_kp_j$$

Sumando $(m+1)w_jp_j$ a ambos lados de la desigualdad

$$(m-1)w_jC_j(\boldsymbol{x})+m\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succeq j}} w_kp_j+\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\preceq j}} w_jp_k \leq 2(m-1)w_jp_j+\sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\preceq j}} w_jp_k+\sum_{t\in\mathcal{M}}\sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succeq j}} w_kp_j$$

Sumando para todo $j\in\mathcal{J}$, y usando (3.3) y (3.6) se tiene:

$$2mC(\boldsymbol{x}) \leq 2\tilde{C}+2(m-1)\sum_{j\in\mathcal{J}} w_jp_j$$

$$mC(\boldsymbol{x}) \leq \tilde{C}+(m-1)\sum_{j\in\mathcal{J}} w_jp_j \tag{3.8}$$

Ordenando

$$C(\boldsymbol{x}) \leq \frac{\tilde{C}}{m}+\frac{m-1}{m}\sum_{j\in\mathcal{J}} w_jp_j = \frac{\tilde{C}}{m}+\frac{m-1}{2m}\sum_{j\in\mathcal{J}} w_jp_j+\frac{m-1}{2m}\sum_{j\in\mathcal{J}} w_jp_j$$

Por (3.2), se tiene que $\sum_{j\in\mathcal{J}} w_jp_j \leq C(\boldsymbol{x}^*)$. Usado esta cota superior para $\sum_{j\in\mathcal{J}} w_jp_j$ y la cota inferior para $C(\boldsymbol{x}^*)$ (Teorema 2.1), se tiene que

$$C(\boldsymbol{x}) \leq C(\boldsymbol{x}^*)+\frac{m-1}{2m}\sum_{j\in\mathcal{J}} w_jp_j \leq C(\boldsymbol{x}^*)\left(\frac{3}{2}-\frac{1}{2m}\right)$$

La cota inferior es establecida por el Lema 3.3. Finalmente, se tiene $\frac{6}{5} \leq \frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} \leq \frac{3}{2}-\frac{1}{2m}$. □

Para el caso donde todas las tareas tienen el mismo cociente de Smith, Brueggemann et al. [15] presentan el siguiente factor de aproximación ajustado.

Teorema 3.5 *(Brueggemann, Hurink y Kern [15]).*
Si todas las tareas tienen el mismo cociente de Smith. Entonces el factor de aproximación de un jump-óptimo del problema $P||\sum w_jC_j$ es $\frac{9-\sqrt{6}}{6} \approx 1{,}092$

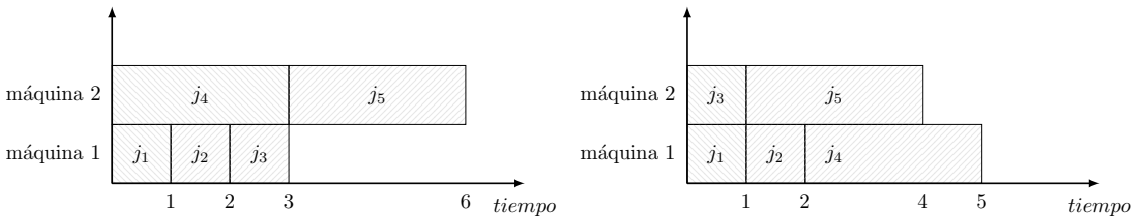
A continuación se presenta un factor de aproximación no-constante para instancias del problema $P||\sum w_jC_j$.

Teorema 3.6 *Sea $\mathcal{I}$ una instancia del problema $P||\sum w_jC_j$. El factor de aproximación de cualquier solución jump-óptima de $\mathcal{I}$ es ρ. Más aún, este factor es ajustado.*

$$\rho \leq 1+\frac{(m-1)\sum_{j\in\mathcal{J}} w_jp_j}{2\tilde{C}+(m-1)\sum_{j\in\mathcal{J}} w_jp_j}$$

Demostración. De (3.8), se tiene que:

$$mC(\boldsymbol{x}) \leq \tilde{C}+(m-1)\sum_{j\in\mathcal{J}} w_jp_j$$



(a) Solución jump-óptima. (b) Solución óptima.

Figura 3.3: Soluciones para instancia de $P||\sum C_j$

Por el Teorema 2.1, se tiene que:

$$mC(\boldsymbol{x}^*) \geq \tilde{C} + \frac{(m-1)}{2}\sum_{j\in\mathcal{J}} w_j p_j$$

Luego,

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \rho \leq 1 + \frac{(m-1)\sum_{j\in\mathcal{J}} w_j p_j}{2\tilde{C} + (m-1)\sum_{j\in\mathcal{J}} w_j p_j}$$

Para demostrar que ρ es ajustado, consideramos la instancia presentada en el Lema 3.3. Para la cual se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \rho = \frac{6}{5}$. □

3.4. Tiempo de completación

En notación de scheduling este problema es representado por $P||\sum C_j$, donde el peso de todas las tareas es unitario. Por el Lema 1.3, se tiene que este problema es equivalente al caso donde el peso es constante para todas las tareas. Para establecer una cota inferior para el factor de aproximación, se presenta una instancia del problema, que expone un peor caso. Luego este resultado es complementado con el Teorema 3.4 para establecer un rango para el factor de aproximación.

Lema 3.7 *El factor de aproximación de una solución jump-óptima para el problema $P||\sum C_j$, es al menos $\frac{15}{13}$.*

Demostración. Sea una instancia con $m = 2$ máquinas y $n = 5$ tareas, con tiempos de proceso $\boldsymbol{p} = (1, 1, 1, 3, 3)$.
El costo de cualquier solución óptima para esta instancia es $C(\boldsymbol{x}^*) = 13$, mientras que el costo de algún peor jump-óptimo tiene un costo de $C(\boldsymbol{x}) = 15$. Luego, el factor de aproximación es $\frac{15}{13}$.
Como las máquinas son idénticas y la instancia tiene algunas tareas con el mismo tiempo de proceso, existen múltiples soluciones óptimas y múltiples soluciones jump-óptimas. Un ejemplo de cada una de estas soluciones es $\boldsymbol{x}^* = (1, 1, 2, 1, 2)$ y $\boldsymbol{x} = (1, 1, 1, 2, 2)$. En la figura 3.3 las tareas j_2 y j_4 en la otra máquina. En la figura 3.2 se presentan cartas Gantt de estas soluciones. □

Teorema 3.8 *El factor de aproximación de una solución jump-óptima para el problema* $P||\sum C_j$*, se encuentra entre* $\frac{15}{13}$ *y* $\frac{3}{2} - \frac{1}{2m}$.

Demostración. El Teorema 3.4 establece una cota superior para el problema $P||\sum w_j C_j$. Como el problema $P||\sum C_j$ es un caso particular del problema $P||\sum w_j C_j$ se establece la cota superior sobre el factor de aproximación. La cota inferior es establecida por el Lema 3.7. □

A continuación se presenta un factor de aproximación no-constante para instancias del problema $P||\sum C_j$.

Teorema 3.9 *Sea* $\mathcal{I}$ *una instancia del problema* $P||\sum C_j$*. El factor de aproximación de cualquier solución jump-óptima de* $\mathcal{I}$ *es* λ*. Más aún, este factor es ajustado.*

$$\lambda \leq \frac{\frac{n+m}{m}\sum_{j=1}^{n} p_j - \sum_{j=1}^{n} \frac{j}{m} p_j}{\frac{n+m}{m}\sum_{j=1}^{n} p_j - \sum_{j=1}^{n} \lceil \frac{j}{m} \rceil p_j}$$

Demostración. Sea n la cantidad de tareas y m la cantidad de máquinas. Sin pérdida de generalidad, se asume que $\frac{n}{m} \in \mathbb{N}$. Si ésto no ocurre, se puede agregar tareas con tiempo de proceso cero. Además se considera que las tareas están ordenadas, tal que $p_1 \leq p_2 \leq \cdots \leq p_n$. El tiempo de completación total en un máquina (3.6) puede ser expresado como $\tilde{C} = \sum_{j=1}^{n} (n - j + 1) p_j$. Usando (3.8), se tiene la siguiente cota superior para $C(\boldsymbol{x})$:

$$C(\boldsymbol{x}) \leq \frac{n+m}{m} \sum_{j=1}^{n} p_j - \sum_{j=1}^{n} \frac{j}{m} p_j \tag{3.9}$$

Por otra parte, para este problema es posible determinar el óptimo global usando la regla SPT, asignando secuencialmente las tareas en el orden SPT a la máquina con menor tiempo de operación. Por el Teorema 2.3, y dado que $p_1 \leq p_2 \leq \cdots \leq p_n$, se tiene que la tarea j será asignada en la posición $\lceil \frac{j}{m} \rceil$-ésima desde el inicio. Luego, el costo del óptimo global es:

$$C(\boldsymbol{x}^*) = \sum_{j=1}^{n} \left(\frac{n}{m} - \left\lceil \frac{j}{m} \right\rceil + 1 \right) p_j \tag{3.10}$$

Por (3.9) y (3.10), se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \lambda$.
Para demostrar que λ es ajustado, consideramos la instancia presentada en el Lema 3.7. Para que se cumpla $\frac{n}{m} \in \mathbb{N}$, es necesario agregar una tarea con tiempo de proceso cero. Con ello, se puede determinar que $\lambda = \frac{15}{13}$. □

3.5. Tiempo ponderado de completación con tareas unitarias

En notación de scheduling este problema es representado por $P|p_j = 1|\sum w_j C_j$, donde el tiempo de proceso de todas las tareas es unitario.

Para determinar cotas para el factor de aproximación de las soluciones jump-óptimas, se demostrará que este problema es equivalente al problema $P||\sum C_j$, presentado en la Sección 3.4. Por lo tanto, las soluciones jump-óptimas del problema $P|p_j = 1|\sum w_j C_j$ tienen el mismo factor de aproximación que las soluciones jump-óptimas del problema $P||\sum C_j$.

Definición 3.10 *Sea $\mathcal{I}$ una instancia de $P||\sum C_j$ con tiempos de proceso $q_1 \leq q_2 \leq \cdots \leq q_n$. Se define la transformada $\Upsilon : \mathcal{I} \to \mathcal{I}'$, tal que $\Upsilon(\mathcal{I})$ genera una instancia $\mathcal{I}'$ del problema $P|p_j = 1|\sum w_j C_j$ con los mismos datos de $\mathcal{I}$. Esto es, para todo $j \in \mathcal{J}$ se hace $w'_j = q_j$ y $p'_j = 1$. Se puede notar que al aplicar la transformada inversa a $\mathcal{I}'$, se obtiene la instancia $\mathcal{I}$.*

Lema 3.11 *Sea $\mathcal{I}$ una instancia de $P||\sum C_j$, $\boldsymbol{z}$ una solución factible de $\mathcal{I}$, y sea $\mathcal{I}'$ la instancia obtenida al aplicar la transformada Υ sobre $\mathcal{I}$. La solución $\boldsymbol{z}$ tiene el mismo costo en ambas instancias, $\mathcal{I}'$ y $\mathcal{I}$.*

Demostración. Para simplificar las expresiones se considerará, sin pérdida de generalidad, que $q_1 < q_2 < \ldots < q_n$. De (3.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} w_j p_k$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}'$.
En la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} q_k$$

Para la instancia $\mathcal{I}$, el secuenciamiento de las tareas en $\boldsymbol{z}$ es determinado por la regla SPT. Por lo tanto, se puede sustituir el símbolo $\preceq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \leq j}} q_k$$

Por otra parte, de (3.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succeq j}} w_k p_j$$

En la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{j \in \mathcal{J}} w_j C_j(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succeq j}} q_k$$

Para la instancia $\mathcal{I}'$, el secuenciamiento es determinado por la regla WSPT. Como en la instancia $\mathcal{I}'$ el tiempo de proceso de cada tarea es unitario, la regla WSPT es equivalente a

secuenciar las tareas en orden decreciente de acuerdo al peso de las tareas ($w_j = q_j$). Por lo tanto, se puede sustituir el símbolo $\succeq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \leq j}} q_k$$

Finalmente, se tiene que $C\left(\mathcal{I}(\boldsymbol{z})\right) = C\left(\mathcal{I}'(\boldsymbol{z})\right)$. □

Lema 3.12 *Sea $\boldsymbol{x}$ una solución jump-óptima de una instancia $\mathcal{I}$ del problema $P||\sum C_j$. Luego, $\boldsymbol{x}$ también es un jump-óptimo para la instancia $\mathcal{I}' = \Upsilon(\mathcal{I})$ del problema $P|p_j = 1|\sum w_j C_j$.*

Demostración. De la condición de optimalidad local (3.7) se tiene

$$w_j C_j(\boldsymbol{x}) + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} w_k p_j \leq w_j p_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \prec j}} w_j p_k + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \succ j}} w_k p_j$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}'$. Para la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$, para todo $j \in \mathcal{J}$. Mientras que para la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$, para todo $j \in \mathcal{J}$.
El objetivo es probar que $\boldsymbol{x}$ satisface la condición de optimalidad local para $\mathcal{I}'$. Es decir:

$$q_j C_j(\mathcal{I}'(\boldsymbol{x})) + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} q_k \leq q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \prec j}} q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \succ j}} q_k \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M} \setminus \{i\}$$

Pero $C_j(\mathcal{I}'(\boldsymbol{x})) = \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \preceq j}} p_k = \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \preceq j}} 1$. Entonces

$$\sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \preceq j}} q_j + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} q_k \leq q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \prec j}} q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \succ j}} q_k \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M} \setminus \{i\}$$

$$\sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \prec j}} q_j + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} q_k \leq \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \prec j}} q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \succ j}} q_k \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M} \setminus \{i\}$$

Para la instancia $\mathcal{I}'$, la secuencia de las tareas es definida por la regla WSPT, pero como las tareas tienen tiempo de proceso unitario, la secuencia de las tareas se define en orden decreciente según el peso de las tareas. Por lo tanto, se puede sustituir el símbolo $\succ$ por $<$ y el símbolo $\prec$ por $>$. Entonces la condición de optimalidad en $\mathcal{I}'$ es:

$$\sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k > j}} q_j + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k < j}} q_k \leq \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k > j}} q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k < j}} q_k \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M} \setminus \{i\} \tag{3.11}$$

Por otra parte, dado que $\boldsymbol{x}$ es un jump-local de $\mathcal{I}$, se sabe que:

$$C_j(\mathcal{I}(\boldsymbol{x})) + \sum_{\substack{k \in N_i(\boldsymbol{x}) \\ k \succ j}} q_j \leq q_j + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \prec j}} q_k + \sum_{\substack{k \in N_t(\boldsymbol{x}) \\ k \succ j}} q_j \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M} \setminus \{i\}$$

Pero $C_j(\mathcal{I}(\boldsymbol{x})) = \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\preceq j}} p_k = \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\preceq j}} q_k$. Entonces

$$\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\preceq j}} q_k + \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} q_j \leq q_j + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} q_j \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M}\setminus\{i\}$$

$$\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k\succ j}} q_j \leq \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k\succ j}} q_j \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M}\setminus\{i\}$$

Para la instancia $\mathcal{I}$, la secuencia de las tareas es definida por la regla SPT. Por lo tanto, se puede sustituir el símbolo $\succ$ por $>$ y el símbolo $\prec$ por $<$. Entonces la condición de optimalidad en $\mathcal{I}$ es:

$$\sum_{\substack{k\in N_i(\boldsymbol{x})\\ k< j}} q_k + \sum_{\substack{k\in N_i(\boldsymbol{x})\\ k> j}} q_j \leq \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k< j}} q_k + \sum_{\substack{k\in N_t(\boldsymbol{x})\\ k> j}} q_j \ , \forall j \in \mathcal{J}, i \in \mathcal{M}, t \in \mathcal{M}\setminus\{i\} \tag{3.12}$$

Finalmente, por (3.11) y (3.12), se tiene que la condición de optimalidad es la misma para ambas instancias. Por lo tanto, si $\boldsymbol{x}$ es un jump-óptimo de la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. □

Teorema 3.13 *El factor de aproximación de una solución jump-óptima para el problema* $P|p_j = 1|\sum w_j C_j$, *se encuentra entre* $\frac{15}{13}$ *y* $\frac{3}{2} - \frac{1}{2m}$.

Demostración. Sea $\mathcal{I}$ una instancia del problema $P||\sum C_j$; $\boldsymbol{x}$ una solución jump-óptima de $\mathcal{I}$; $\mathcal{I}'$ la instancia del problema $P|p_j = 1|\sum w_j C_j$ obtenida al aplicar la transformada Υ sobre $\mathcal{I}$.
Por el Lema 3.12, si $\boldsymbol{x}$ es un jump-óptimo para la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. Por el Lema 3.11, se tiene que el costo de cualquier solución es el mismo en ambas instancias, $\mathcal{I}$ y $\mathcal{I}'$. Luego, se tiene que para cualquier instancia $\mathcal{I}'$ el factor de aproximación es el mismo que se tiene para la instancia $\mathcal{I}$. Entonces, el problema $P|p_j = 1|\sum w_j C_j$ tiene el mismo factor de aproximación que el problema $P||\sum C_j$, el cual es establecido por el Teorema 3.8. □

Capítulo 4

Máquinas paralelas idénticas restringidas

En este capítulo se presentan resultados que permiten determinar la calidad de los óptimos locales para el problema de minimizar el tiempo ponderado de completación en máquinas paralelas idénticas restringidas $RP||\sum w_jC_j$. Además del caso general, se consideran los casos: no ponderado $RP||\sum C_j$, tiempo de proceso constante ponderado $RP|p_j=1|\sum w_jC_j$ y no ponderado $RP|p_j=1|\sum C_j$. Las estructuras de vecindario analizadas son jump y swap.

Para el problema $RP||\sum w_jC_j$ se determinó que el factor de aproximación de las soluciones jump-óptimas y swap-óptimas se encuentra entre 1,75 y 1,809. Para los problemas $RP||\sum C_j$ y $RP|p_j=1|\sum w_jC_j$, estos factores se encuentran entre 1,533 y 1,618. Finalmente, para el problema $RP|p_j=1|\sum C_j$ se presenta una conjetura que indica que el factor de aproximación para ambos vecindarios es 1,533.

4.1. Introducción

Para describir el problema y sus variantes se define la siguiente notación:

$\mathcal{J}=\{1,\ldots,n\}$: el conjunto tareas;

$\mathcal{M}=\{1,\ldots,m\}$: el conjunto de máquinas;

$\mathcal{M}_j\subseteq\mathcal{M}$: conjunto de máquinas que puede procesar la tarea j;

p_j: tiempo necesario para procesar la tarea $j\in\mathcal{J}$;

w_j: peso o importancia relativa de la tarea $j\in\mathcal{J}$;

$\boldsymbol{z}$: una solución factible del problema;

$C_j(\boldsymbol{z})$: tiempo de completación de la tarea j en la solución $\boldsymbol{z}$;

$C(\boldsymbol{z})=\sum_{j\in\mathcal{J}}w_jC_j(\boldsymbol{z})$: costo de la solución $\boldsymbol{z}$;

$N_i(\boldsymbol{z})$: conjunto de tareas asignadas a la máquina i en la solución $\boldsymbol{z}$;

En este ambiente de máquinas, cada tarea $j \in \mathcal{J}$ puede ser procesada, sin interrupción, por un conjunto restringido de máquinas $\mathcal{M}_j \subseteq \mathcal{M}$, requiriendo un tiempo de proceso p_j. Cada máquina puede procesar una tarea a la vez, y todas las tareas y máquinas están disponibles desde el inicio. El problema consiste en determinar una solución $\boldsymbol{z}$, que minimice $C(\boldsymbol{z})$. La solución $\boldsymbol{z}$, establece la máquina a la que cada tarea es asignada y la secuencia en que las tareas son procesadas. Para cada máquina, la secuencia es definida por las reglas WSPT y SPT, para los objetivos $\sum w_j C_j$ y $\sum C_j$ respectivamente (Teoremas 1.1 y 1.2).

Una instancia $\mathcal{I}$ del problema, queda definida por los conjuntos $\mathcal{J}$, $\mathcal{M}$ y $\mathcal{M}_j$, $\forall j \in \mathcal{J}$; y las características de las tareas. Donde $\boldsymbol{w}$ y $\boldsymbol{p}$ representan el vector de pesos y tiempos de proceso de las tareas respectivamente.

Sea $\boldsymbol{z}$ cualquier solución factible de la instancia $\mathcal{I}$. El costo de la solución $\boldsymbol{z}$ puede ser determinado por (4.1), donde el símbolo $k \prec j$ indica que la tarea k precede a la tarea j. Por otra parte $k \succ j$ indica que la tarea k sucede a la tarea j (los símbolos $\preceq$ y $\succeq$ además incluyen la tarea j). La precedencia entre las tareas se define por la regla WSPT o SPT según corresponda. En los casos donde exista empate entre dos o más tareas, su precedencia se define en forma arbitraria.

$$C(\boldsymbol{z}) = \sum_{j \in \mathcal{J}} w_j p_j + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \prec j}} w_j p_k = \sum_{j \in \mathcal{J}} w_j p_j + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succ j}} w_k p_j \tag{4.1}$$

En notación de scheduling el problema es representado por $RP||\sum w_j C_j$ o $P|\mathcal{M}_j|\sum w_j C_j$. A partir de este problema general, es posible obtener algunos casos particulares (las equivalencias quedan demostradas por el Lema 1.3):

1. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son arbitrarios, se tiene el problema $RP||\sum w_j C_j$
2. Si $\boldsymbol{w}$ es constante, el problema resultante es equivalente a $RP||\sum C_j$
3. Si $\boldsymbol{p}$ es constante, el problema resultante es equivalente a $RP|p_j = 1|\sum w_j C_j$
4. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son constantes, el problema resultante es equivalente a $RP|p_j = 1|\sum C_j$

En las siguientes secciones se analiza la calidad de los óptimos locales para vecindarios de jump y swap para los cuatro casos. Para determinar el factor de aproximación de los óptimos locales, se determinaron cotas superiores e inferiores para estos factores de aproximación. Para establecer las cotas superiores se utilizó, el mapeo $\varphi : \mathcal{M}^{\mathcal{J}} \to L_2\left([0,\infty)\right)^{\mathcal{M}}$ propuesto por Cole et al. [21], la reducción de elegibilidad de máquina y las desigualdades presentadas en la Sección 2.3. Para determinar las cotas inferiores se presentan instancias que exponen un peor caso para los problemas.

4.2. Tiempo ponderado de completación

Para cualquier instancia $\mathcal{I}$, la asignación óptima de tareas a las máquinas es representada por el vector $\boldsymbol{x}^*$. Donde x_j^* indica la máquina donde es asignada la tarea j, es decir, $x_j^* = i$ si la tarea j es programada en la máquina i. Adicionalmente se define:

$N_i(\boldsymbol{x}^*) = \{j \in \mathcal{J} : x_j^* = i\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}^*$;

$C_j(\boldsymbol{x}^*)$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}^*$;

$C(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x}^*)$: el costo óptimo.

La solución jump-óptima es representada por el vector $\boldsymbol{x}$. Donde x_j indica la máquina donde es asignada la tarea j, es decir, $x_j = i$ si la tarea j es programada en la máquina i en el óptimo local. Adicionalmente se define:

$N_i(\boldsymbol{x}) = \{j \in \mathcal{J} : x_j = i\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}$;

$C_j(\boldsymbol{x})$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}$;

$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x})$: el costo de la solución jump-óptima $\boldsymbol{x}$.

Sea $\eta = \sum_{j\in\mathcal{J}} w_j p_j$. Con lo anterior y (4.1), es posible determinar los costos de las soluciones óptima y jump-óptima como:

$$C(\boldsymbol{x}^*) = \eta + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\prec j}} w_j p_k = \eta + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\succ j}} w_k p_j \tag{4.2}$$

$$C(\boldsymbol{x}) = \eta + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\prec j}} w_j p_k = \eta + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k p_j \tag{4.3}$$

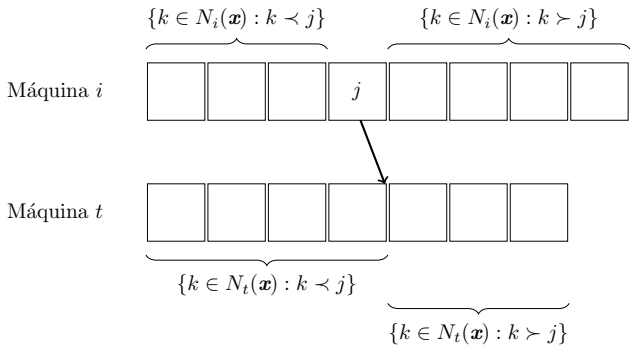


Figura 4.1: Esquema para movida de jump en $RP||\sum w_j C_j$

En la figura 4.1 se presenta un esquema de la movida de jump para la tarea j. Al asignar la tarea j a la máquina t, se produce un cambio en su tiempo de completación, este será la

suma entre su tiempo de proceso (p_j) y el tiempo de proceso de todas las tareas asignadas a la máquina t que preceden a la tarea j. Al retirar la tarea j de la máquina i, el tiempo de completación de todas las tareas que suceden a la tarea j se reduce en p_j. Al asignar la tarea j a la máquina t, el tiempo de completación de todas las tareas que suceden a la tarea j en la máquina t aumenta en p_j.

Lema 4.1 *El factor de aproximación de una solución jump-óptima para el problema* $RP||\sum w_jC_j$, *es a lo más* $1+\frac{\phi}{2} \approx 1{,}809$.

Demostración. Por el Lema 2.6, al aplicar la reducción de elegibilidad de máquina, se tiene que cada tarea $j \in \mathcal{J}$ podrá ser procesada por las máquinas x_j y x_j^*, sin alterar el factor de aproximación. Esto es $\mathcal{M}'_j = \{x_j, x_j^*\}$ para todo $j \in \mathcal{J}$. Sea δ_j el ahorro en $C(\boldsymbol{x})$ que se obtiene al retirar la tarea j de la máquina x_j; y sea δ'_j el incremento en $C(\boldsymbol{x})$ al asignar la tarea j a la máquina x_j^*.

$$\delta_j = w_jC_j(\boldsymbol{x}) + p_j \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k \tag{4.4}$$

$$\delta'_j = w_jp_j + w_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} p_k + p_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} w_k \tag{4.5}$$

Para que $\boldsymbol{x}$ sea un jump-óptimo, se debe cumplir la condición $\delta_j \le \delta'_j$ para todo $j \in \mathcal{J}$. Por lo tanto, la condición de optimalidad local para el vecindario de jump es:

$$w_jC_j(\boldsymbol{x}) + p_j \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k \le w_jp_j + w_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} p_k + p_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} w_k \tag{4.6}$$

Sumando w_jp_j a ambos lados

$$w_jC_j(\boldsymbol{x}) + w_jp_j + p_j \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k \le 2w_jp_j + w_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} p_k + p_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} w_k$$

Agrupando las últimas dos componentes y considerando que: si $k \prec j$, entonces $\frac{p_k}{w_k} \le \frac{p_j}{w_j}$; si $k \succ j$, entonces $\frac{p_j}{w_j} \le \frac{p_k}{w_k}$.

$$w_jC_j(\boldsymbol{x}) + w_jp_j + p_j \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k \le 2w_jp_j + \sum_{k\in N_{x_j^*}(\boldsymbol{x})} w_jw_k \min\left\{\frac{p_k}{w_k}, \frac{p_j}{w_j}\right\}$$

Sumando para todo $j \in \mathcal{J}$ y usando $\eta = \sum_{j\in\mathcal{J}} w_jp_j$

$$\sum_{j\in\mathcal{J}} w_jC_j(\boldsymbol{x}) + \eta + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_kp_j \le 2\eta + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{x}^*)} \sum_{k\in N_i(\boldsymbol{x})} w_jw_k \min\left\{\frac{p_k}{w_k}, \frac{p_j}{w_j}\right\}$$

Usando (4.3), se tiene

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{x}^*)} \sum_{k\in N_i(\boldsymbol{x})} w_j w_k \,\text{mín}\left\{\frac{p_k}{w_k}, \frac{p_j}{w_j}\right\}$$

Por (2.5)

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy \tag{4.7}$$

Aplicando la desigualdad del Lema 2.9

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i\in\mathcal{M}} \int_0^\infty \left(\frac{\beta}{2} f_i^{*2}(y) + \frac{1}{2\beta} f_i^2(y)\right) dy$$

Ordenando y usando (2.4)

$$2C(\boldsymbol{x}) \leq 2\eta + \frac{\beta}{2}\left(2C(\boldsymbol{x}^*) - \eta\right) + \frac{1}{2\beta}\left(2C(\boldsymbol{x}) - \eta\right)$$

$$\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \leq \beta C(\boldsymbol{x}^*) + \left(2 - \frac{\beta}{2} - \frac{1}{2\beta}\right)\eta$$

Suponiendo que:

$$2 - \frac{1}{\beta} \geq 0 \tag{4.8}$$

$$2 - \frac{\beta}{2} - \frac{1}{2\beta} \geq 0 \tag{4.9}$$

De (4.2), se tiene que $\eta \leq C(\boldsymbol{x}^*)$. Entonces:

$$\left(2 - \frac{1}{\beta}\right) C(\boldsymbol{x}) \leq C(\boldsymbol{x}^*)\left(2 + \frac{\beta}{2} - \frac{1}{2\beta}\right)$$

Haciendo $\beta = \phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618$ (número áureo), se cumple (4.8) y (4.9), y se alcanza el mayor valor para $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)}$. Usando $1 + \phi = \phi^2$, se tiene:

$$(3 - \phi) C(\boldsymbol{x}) \leq \frac{5}{2} C(\boldsymbol{x}^*)$$

Finalmente se tiene

$$C(\boldsymbol{x}) \leq \left(\frac{5}{6 - 2\phi}\right) C(\boldsymbol{x}^*) = \left(1 + \frac{\phi}{2}\right) C(\boldsymbol{x}^*)$$

□

Para establecer una cota inferior para el factor de aproximación de las soluciones jump-óptimas y swap-óptimas, se presenta una instancia del problema que expone un peor caso. La construcción de esta instancia se basa en los resultados presentados por Caragiannis et al. [18] para juegos de balanceo descentralizado de carga de trabajo en máquinas paralelas idénticas restringidas.

Lema 4.2 *El factor de aproximación de una solución jump-óptima o swap-óptima para el problema* $RP||\sum w_jC_j$, *es al menos* 1,75.

Demostración. Sea $\mathcal{I}$ una instancia con $2k+1$ niveles de tareas y $2k+2$ grupos de máquinas. En la figura 4.2 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen del arco k indica la máquina en la cual es asignada la tarea k en la solución óptima. En cambio, el destino del arco k indica la máquina en la cual es asignada la tarea k en la solución jump-óptima. Sea j_t alguna tarea del nivel t, $t = 1, \ldots, 2k+1$. Las tareas del nivel t se pueden procesar solamente en dos máquinas, una de ellas pertenece al grupo $t-1$ y otra pertenece al grupo t. En el jump-óptimo $\boldsymbol{x}$, las máquinas del grupo $t-1$ procesan las tareas j_t. Mientras que en el óptimo global $\boldsymbol{x}^*$, estas son procesadas por las máquinas del grupo t. Dentro de cada nivel, las tareas son idénticas. Además, el peso es igual al tiempo de proceso. Estos se definen como:

$$w_{j_t} = p_{j_t} = \begin{cases} (2/3)^t & , \ \forall t = 1, \ldots, k \\ (2/3)^k (1/2)^{t-k-1} & , \ \forall t = k+1, \ldots, 2k \\ 2(1/3)^k & , \ t = 2k+1 \end{cases}$$

La cantidad de máquinas en cada grupo es:

$$q_t = \begin{cases} 3^t & , \ \forall t = 0, \ldots, k \\ 3^k 2^{t-k} & , \ \forall t = k+1, \ldots, 2k \\ 6^k & , \ t = 2k+1 \end{cases}$$

En el óptimo global $\boldsymbol{x}^*$, todas las máquinas procesan una tarea, excepto las máquinas del grupo 0, las que no procesan ninguna tarea. En el jump-óptimo $\boldsymbol{x}$, las máquinas de los grupos $i = 0, \ldots, k-1$ procesan tres tareas, las máquinas de los grupos $i = k, \ldots, 2k-1$ procesan dos tareas, las máquinas del grupo $2k$ procesan una tarea, mientras que las máquinas del grupo $2k+1$ no procesan tareas.
Para verificar que $\boldsymbol{x}$ es un óptimo local de $\mathcal{I}$, se usarán las expresiones (4.4) y (4.5), de tal manera que se cumpla la condición de optimalidad local (4.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}$.
Para $t = 1, \ldots, k-1$, mover alguna tarea j_t desde alguna máquina perteneciente al grupo $t-1$, a alguna máquina elegible en el grupo t, genera un ahorro $\delta_{j_t} = 3w_{j_t}p_{j_t}$, y un incremento $\delta'_{j_t} = w_{j_t}\left(3p_{j_{t+1}} + p_{j_t}\right)$. Pero $p_{j_{t+1}} = (2/3)p_{j_t}$, entonces $\delta_{j_t} = \delta'_{j_t}$. Por lo tanto, no se realizan movidas para tareas $j_t, \ t = 1, \ldots, k-1$.
Para $t = k$, mover alguna tarea j_k desde alguna máquina perteneciente al grupo $k-1$, a alguna máquina elegible en el grupo k, genera un ahorro $\delta_{j_k} = 3w_{j_k}p_{j_k}$, y un incremento $\delta'_{j_k} = w_{j_k}\left(2p_{j_{k+1}} + p_{j_k}\right)$. Pero $p_{j_{k+1}} = p_{j_k}$, entonces $\delta_{j_k} = \delta'_{j_k}$. Por lo tanto, no se realizan movidas para las tareas j_k.
Para $t = k+1, \ldots, 2k-1$, mover alguna tarea j_t desde alguna máquina perteneciente al grupo $t-1$, a alguna máquina elegible en el grupo t, genera un ahorro $\delta_{j_t} = 2w_{j_t}p_{j_t}$, y un incremento $\delta'_{j_t} = w_{j_t}\left(2p_{j_{t+1}} + p_{j_t}\right)$. Pero $p_{j_{t+1}} = (1/2)p_{j_t}$, entonces $\delta_{j_t} = \delta'_{j_t}$. Por lo tanto, no se realizan movidas para tareas $j_t, \ t = k+1, \ldots, 2k-1$.
Para $t = 2k$, mover alguna tarea j_{2k} desde alguna máquina perteneciente al grupo $2k-1$, a alguna máquina elegible en el grupo $2k$, genera un ahorro $\delta_{j_{2k}} = 2w_{j_{2k}}p_{j_{2k}}$, y un incremento

$\delta'_{j_{2k}} = w_{j_{2k}} \left(p_{j_{2k+1}} + p_{j_{2k}}\right)$. Pero $p_{j_{2k+1}} = p_{j_{2k}}$, entonces $\delta_{j_{2k}} = \delta'_{j_{2k}}$. Por lo tanto, no se realizan movidas para las tareas j_{2k}.
Finalmente, las tareas j_{2k+1} tienen el mismo tiempo de proceso en las máquinas del grupo $2k$ y $2k+1$. Por lo tanto, realizar una movida para estas tareas no genera una reducción en $C(\boldsymbol{x})$.

Los costos de las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$ son:

$$\begin{aligned}
C(\boldsymbol{x}) &= 6\sum_{t=1}^{k} q_{t-1}w_t p_t + 3\sum_{t=k+1}^{2k} q_{t-1}w_t p_t + q_{2k}w_{2k+1}p_{2k+1} \\
&= 6\sum_{t=1}^{k} 3^{t-1}\left((2/3)^t\right)^2 + 3\sum_{t=k+1}^{2k} 3^k 2^{t-k-1}\left((2/3)^k(1/2)^{t-k-1}\right)^2 + 6^k\left(2(1/3)^k\right)^2 \\
&= (8/3)\sum_{t=0}^{k-1}(4/3)^t + 3\,(4/3)^k\sum_{t=0}^{k-1}(1/2)^t + 4(2/3)^k \\
&= 14(4/3)^k - 8 - 2(2/3)^k
\end{aligned}$$

$$\begin{aligned}
C(\boldsymbol{x}^*) &= \sum_{t1}^{2k+1} q_{t-1}w_t p_t \\
&= \sum_{t1}^{k} q_t w_t p_t + \sum_{t=k+1}^{2k} q_t w_t p_t + q_{2k+1}w_{2k+1}p_{2k+1} \\
&= \sum_{t=1}^{k} 3^t\left((2/3)^t\right)^2 + \sum_{t=k+1}^{2k} 3^k 2^{t-k}\left((2/3)^k(1/2)^{t-k-1}\right)^2 + 6^k\left(2(1/3)^k\right)^2 \\
&= \sum_{t=1}^{k}(4/3)^t + 2(4/3)^k\sum_{t=0}^{k-1}(1/2)^t + 4(2/3)^k \\
&= 8(4/3)^k - 4
\end{aligned}$$

Luego, el factor de aproximación para la solución jump-óptima $\boldsymbol{x}$ es:

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{14(4/3)^k - 8 - 2(2/3)^k}{8(4/3)^k - 4} = \frac{7 - 4(3/4)^k - (1/2)^k}{4 - 2(3/4)^k}$$

Finalmente, para k suficientemente grande y $\varepsilon > 0$, se tiene que $\dfrac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \dfrac{7}{4} - \varepsilon = 1{,}75 - \varepsilon$.

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}$ tal que $j \neq k$, se tiene que $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo.

□

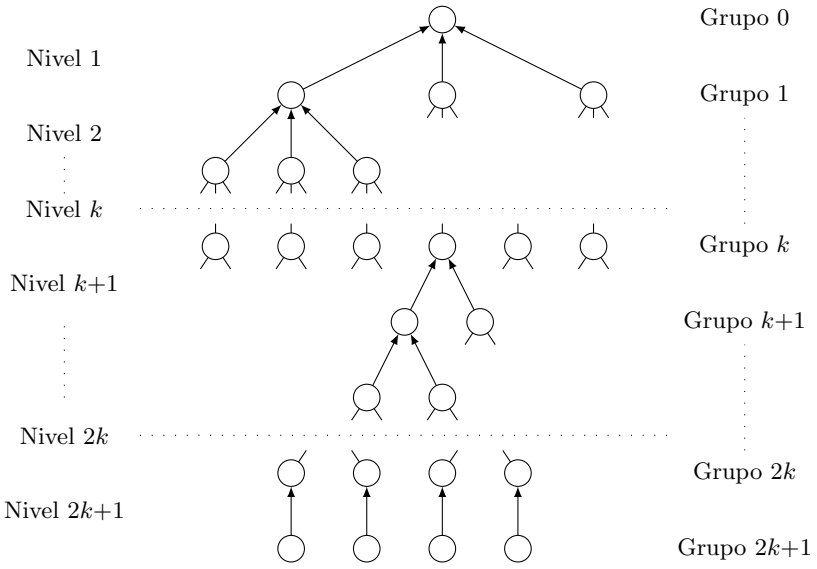


Figura 4.2: Instancia peor caso $RP||\sum w_jC_j$

Con los Lemas 4.1 y 4.2, se puede determinar cotas para el factor de aproximación de las soluciones jump-óptima y swap-óptima, Teoremas 4.3 y 4.4 respectivamente.

Teorema 4.3 *El factor de aproximación de una solución jump-óptima para el problema* $RP||\sum w_jC_j$*, se encuentra entre* 1,75 *y* 1,809*.*

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 4.1 establece una cota superior para el factor de aproximación. Por otra parte, el Lema 4.2 establece una cota inferior para el factor de aproximación. Por lo tanto, $1{,}75 \leq \varphi^J \leq 1{,}809$. □

Teorema 4.4 *El factor de aproximación de una solución swap-óptima para el problema* $RP||\sum w_jC_j$*, se encuentra entre* 1,75 *y* 1,809*.*

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 4.1 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 1{,}809$. Por otra parte, el Lema 4.2 establece una cota inferior para el factor de aproximación, ya que la instancia presentada es también un swap-óptimo. Luego, $\varphi^S \geq 1{,}75$. Por lo tanto, $1{,}75 \leq \varphi^S \leq 1{,}809$. □

4.3. Tiempo de completación

En notación de scheduling este problema es representado por $RP||\sum C_j$, donde el peso de todas las tareas es unitario. Por el Lema 1.3, se tiene que este problema es equivalente al caso donde el peso es constante para todas las tareas, $w_j = w$ para todo $j \in \mathcal{J}$.

Para este problema, el mapeo $\varphi : \mathcal{M}^{\mathcal{J}} \to L_2([0,\infty))^{\mathcal{M}}$ propuesto por Cole et al. [21], es ligeramente distinto. De (2.2), para $i \in \mathcal{M}_j$ se tiene que:

$$f_i(y) = \sum_{j \in N_i(\boldsymbol{z}): p_j \geq y} 1$$

Como $f_i(y)$ es un contador de tareas asignadas a la máquina i con tiempo de proceso superior o igual a y, se puede imponer condiciones de integralidad sobre los elementos $f_i(y)$.

Lema 4.5 *El factor de aproximación de una solución jump-óptima para el problema $RP||\sum C_j$, es a lo más $\phi \approx 1{,}618$.*

Demostración. De (4.7) se tiene

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy$$

Por la integralidad de los $f_i(y)$ y $f_i^*(y)$, se puede usar la desigualdad del Lema 2.8. Entonces:

$$2C(\boldsymbol{x}) \leq 2\eta + \sum_{i \in \mathcal{M}} \int_0^\infty \left\{ \left(1 - \frac{\phi}{2}\right) f_i^2(y) + \left(1 + \frac{1}{2\phi}\right) f_i^{*2}(y) + \frac{\phi}{2} f_i(y) - \left(1 + \frac{1}{2\phi}\right) f_i^*(y) \right\} dy$$

Para este problema, se tiene que $p_{ij} = p_j$ y $w_j = 1$, para todo $j \in \mathcal{J}$. Por (2.3), se tiene que $\sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) dy = \sum_{i \in \mathcal{M}} \int_0^\infty f_i^*(y) dy = \sum_{j \in \mathcal{J}} p_j = \eta$
Ordenando y usando (2.4) se tiene:

$$2C(\boldsymbol{x}) \leq 2\eta + \left(1 - \frac{\phi}{2}\right)(2C(\boldsymbol{x}) - \eta) + \left(1 + \frac{1}{2\phi}\right)(2C(\boldsymbol{x}^*) - \eta) + \frac{\phi}{2}\eta - \left(1 + \frac{1}{2\phi}\right)\eta$$

Agrupando

$$\phi C(\boldsymbol{x}) \leq (\phi - 1 - \phi^{-1})\eta + (2 + \phi^{-1}) C(\boldsymbol{x}^*)$$

Como $\phi = \frac{1+\sqrt{5}}{2}$ es el número áureo, se tiene que $\phi^{-1} = \phi - 1$ y $\phi^2 = 1 + \phi$. Entonces:

$$\phi C(\boldsymbol{x}) \leq \phi^2 C(\boldsymbol{x}^*)$$

Finalmente, se tiene que $C(\boldsymbol{x}) \leq \phi C(\boldsymbol{x}^*)$. □

A continuación, se presenta una instancia del problema $RP||\sum C_j$, que expone un peor caso para los vecindarios de jump y swap.

Lema 4.6 *El factor de aproximación de una solución jump-óptima o swap-óptima para el problema $RP||\sum C_j$, es al menos* 1,533.

Demostración. Sea $\mathcal{I}$ una instancia con $n = 1126$ tareas y $m = 1009$ máquinas. Cada tarea j tiene tiempo de proceso unitario y $|\mathcal{M}_j| \leq 2$, es decir, a lo más existen dos máquinas en $\mathcal{M}$ que pueden procesar la tarea j.

En la figura 4.3 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen

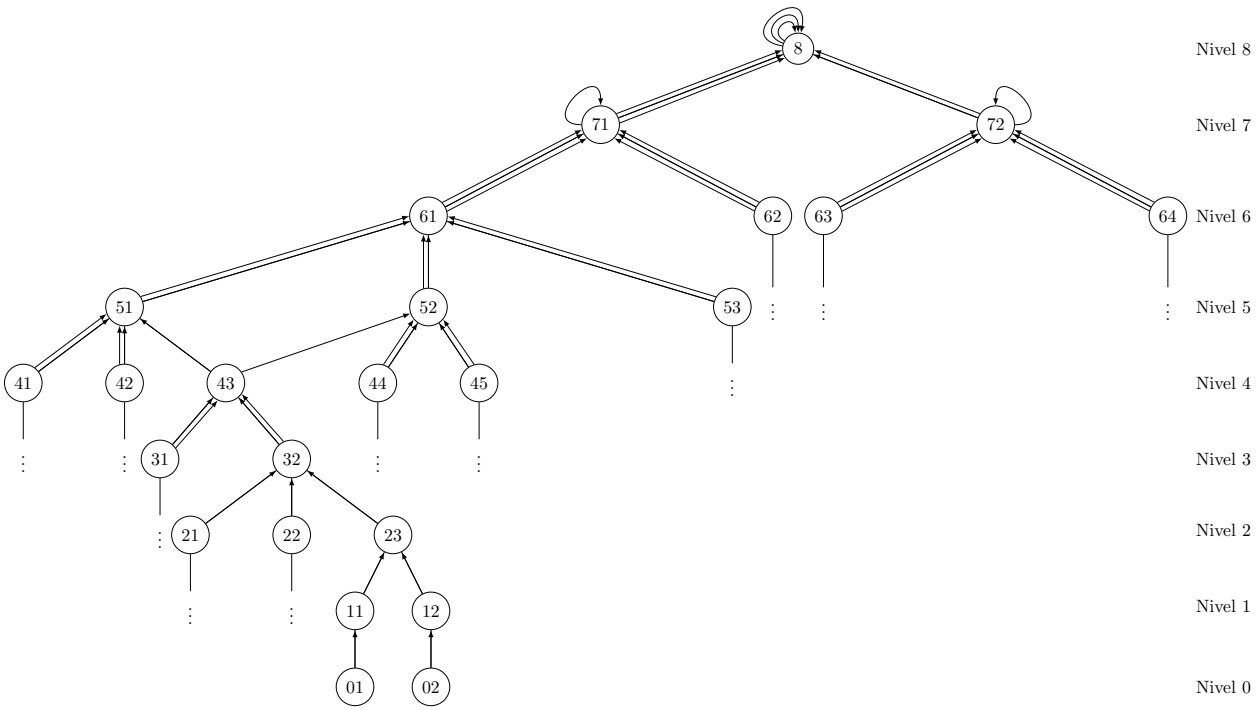


Figura 4.3: Instancia peor caso $RP||\sum C_j$

Nivel	m_k	Tareas en $\boldsymbol{x}$	Aporte a $\boldsymbol{x}$	Tareas en $\boldsymbol{x}^*$	Aporte a $\boldsymbol{x}^*$
8	1	8	36	3	6
7	2	7	56	3 y 4	16
6	4	6	84	3	24
5	12	5	180	2	36
4	30	4	300	2	90
3	60	3	360	2	180
2	180	2	540	1	180
1	360	1	360	1	360
0	360	0	0	1	360
			1916		**1252**

Tabla 4.1: Instancia peor caso $RP||\sum C_j$

del arco k indica la máquina en la cual es asignada la tarea k en la solución óptima $\boldsymbol{x}^*$. En cambio, el destino del arco k indica la máquina en la cual es asignada la tarea k en la solución jump-óptima $\boldsymbol{x}$.

En la tabla 4.1 se presenta un resumen con los datos relevantes de la instancia $\mathcal{I}$, donde m_k es la cantidad de máquinas pertenecientes al nivel k. En la solución jump-óptima, las máquinas pertenecientes al nivel k procesan k tareas. Luego, cada máquina perteneciente al nivel k

aporta T_k al costo del óptimo local $C(\boldsymbol{x})$, donde T_k es el k-ésimo número triangular[1]. Por lo tanto:

$$C(\boldsymbol{x}) = \sum_{k=1}^{8} m_k T_k = 1916$$

Para que $\boldsymbol{x}$ sea un óptimo local de $\mathcal{I}$, es necesario que se cumpla la condición de optimalidad local. Pero dado que el tiempo de proceso de todas las tareas es unitario, sólo basta con determinar la cantidad de tareas asignadas a cada máquina. De acuerdo a lo presentado en la figura 4.3, sea j_k una tarea que puede ser asignada a una máquina perteneciente al nivel k u otra perteneciente al nivel $k-1$. Sean l_k y l_{k-1} esas máquinas respectivamente. Las máquinas l_k y l_{k-1} procesan k y $k-1$ tareas respectivamente. El aporte de ambas al costo del óptimo local es $T_k + T_{k-1}$.
Si se realiza la movida para la tarea k, es decir, es retirada de la máquina l_k y asignada a la máquina l_{k-1}, se tiene que las máquinas l_k y l_{k-1} procesarán $k-1$ y k tareas respectivamente. El aporte de ambas al costo del óptimo local es $T_{k-1}+T_k$. Luego, para $j_k,\ k = 1, \ldots, 8$; realizar una movida no mejora el costo de la solución. Por lo tanto, $\boldsymbol{x}$ es un óptimo local.
En la solución $\boldsymbol{x}^*$, se tiene que las máquinas pertenecientes a los niveles 0, 1 y 2 procesan una tarea en el óptimo global. Por lo tanto, su aporte al costo óptimo $C(\boldsymbol{x}^*)$ es T_1. Las máquinas pertenecientes a los niveles 3, 4 y 5 procesan dos tareas en el óptimo global. Por lo tanto, su aporte a $C(\boldsymbol{x}^*)$ es T_2. Las máquinas pertenecientes a los niveles 6 y 8 procesan tres tareas en el óptimo global. Por lo tanto, su aporte a $C(\boldsymbol{x}^*)$ es T_3. En el nivel 7 hay dos máquinas, las cuales procesan tres y cuatro tareas, por lo que el aporte de ambas es $T_3 + T_4$. Por lo tanto, el costo óptimo es:

$$C(\boldsymbol{x}^*) = (m_0 + m_1 + m_2)T_1 + (m_3 + m_4 + m_5)T_2 + (m_6 + m_8)T_3 + T_3 + T_4 = 1252$$

Finalmente, se tiene que el factor de aproximación para la solución jump-óptima $\boldsymbol{x}$ de la instancia $\mathcal{I}$ es $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = 1{,}5303$

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}$ tal que $j \neq k$, se tiene que $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo. □

Continuando con la idea presentada en la figura 4.3. Es posible construir una instancia de mayor tamaño, permitiendo determinar factores de aproximación cercanos a 1,533. En la Sección 4.5 se presentará un resultado relacionado con este factor de aproximación.

Teorema 4.7 *El factor de aproximación de una solución jump-óptima para el problema* $RP||\sum C_j$, *se encuentra entre* 1,533 *y* 1,618.

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 4.5 establece una cota superior para el factor de aproximación. Por otra parte, el Lema 4.6 establece una cota inferior para el factor de aproximación. Por lo tanto, $1{,}533 \leq \varphi^J \leq 1{,}618$. □

[1] $T = \{1, 3, 6, 10, 15, 21, 28, 36, ...\}$

Teorema 4.8 *El factor de aproximación de una solución swap-óptima para el problema* $RP||\sum C_j$, *se encuentra entre* 1,533 *y* 1,618.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 4.5 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 1{,}618$. Por otra parte, el Lema 4.6 establece una cota inferior para el factor de aproximación, ya que la instancia presentada es también un swap-óptimo. Luego, $\varphi^S \geq 1{,}533$. Por lo tanto, $1{,}533 \leq \varphi^S \leq 1{,}618$. □

4.4. Tiempo ponderado de completación con tareas unitarias

En notación de scheduling este problema es representado por $RP|p_j = 1|\sum w_j C_j$, donde el tiempo de proceso de todas las tareas es unitario.

Para determinar cotas para el factor de aproximación de las soluciones jump-óptimas se demostrará que este problema es equivalente al problema $RP||\sum C_j$ presentado en la Sección 4.3. Por lo tanto, las soluciones jump-óptimas del problema $RP|p_j = 1|\sum w_j C_j$ tienen el mismo factor de aproximación del problema $RP||\sum C_j$. Luego, estos resultados son extendidos a las soluciones swap-óptimas.

Definición 4.9 *Sea* $\mathcal{I}$ *una instancia de* $RP||\sum C_j$ *con tiempos de proceso* $q_1 \leq q_2 \leq \cdots \leq q_n$. *Se define la transformada* $\Upsilon : \mathcal{I} \to \mathcal{I}'$, *tal que* $\Upsilon(\mathcal{I})$ *genera una instancia* $\mathcal{I}'$ *del problema* $RP|p_j = 1|\sum w_j C_j$ *con los mismos datos de* $\mathcal{I}$. *Esto es, para todo* $j \in \mathcal{J}$ *se hace* $w'_j = q_j$ *y* $p'_j = 1$. *Se puede notar que al aplicar la transformada inversa a* $\mathcal{I}'$, *se obtiene la instancia* $\mathcal{I}$.

Lema 4.10 *Sea* $\mathcal{I}$ *una instancia de* $RP||\sum C_j$, $\boldsymbol{z}$ *una solución factible de* $\mathcal{I}$, *y sea* $\mathcal{I}'$ *la instancia obtenida al aplicar la transformada* Υ *sobre* $\mathcal{I}$. *La solución* $\boldsymbol{z}$ *tiene el mismo costo en ambas instancias,* $\mathcal{I}'$ *y* $\mathcal{I}$.

Demostración. Para simplificar las expresiones se considerará, sin pérdida de generalidad, que $q_1 < q_2 < \ldots < q_n$. De (4.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} w_j p_k$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}'$.
En la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{j \in \mathcal{J}} C_j(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} q_k$$

Para la instancia $\mathcal{I}$, el secuenciamiento de las tareas en $\boldsymbol{z}$ es determinado por la regla SPT. Por lo tanto, se puede sustituir el símbolo $\preceq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\leq j}} q_k$$

Por otra parte, de (4.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succeq j}} w_k p_j$$

En la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succeq j}} q_k$$

Para la instancia $\mathcal{I}'$, el secuenciamiento es determinado por la regla WSPT. Como en la instancia $\mathcal{I}'$ el tiempo de proceso de cada tarea es unitario, la regla WSPT es equivalente a secuenciar las tareas en orden decreciente de acuerdo al peso de las tareas ($w_j = q_j$). Por lo tanto, se puede sustituir el símbolo $\succeq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\leq j}} q_k$$

Finalmente, se tiene que $C\left(\mathcal{I}(\boldsymbol{z})\right) = C\left(\mathcal{I}'(\boldsymbol{z})\right)$. □

Lema 4.11 *Sea $\boldsymbol{x}$ una solución jump-óptima de una instancia $\mathcal{I}$ del problema $RP||\sum C_j$. Luego, $\boldsymbol{x}$ también es un jump-óptimo para la instancia $\mathcal{I}' = \Upsilon(\mathcal{I})$ del problema $RP|p_j = 1|\sum w_j C_j$.*

Demostración. De la condición de optimalidad local (4.6) se tiene

$$w_j C_j(\boldsymbol{x}) + p_j \sum_{\substack{k\in N_{x_j}(\boldsymbol{x}) \\ k\succ j}} w_k \leq w_j p_j + w_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}) \\ k\prec j}} p_k + p_j \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}) \\ k\succ j}} w_k \ , \forall j \in \mathcal{J}$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}'$. Para la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$, para todo $j \in \mathcal{J}$. Mientras que para la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$, para todo $j \in \mathcal{J}$.
El objetivo es probar que $\boldsymbol{x}$ satisface la condición de optimalidad local para $\mathcal{I}'$. Es decir:

$$q_j C_j(\mathcal{I}'(\boldsymbol{x})) + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x}) \\ k\succ j}} q_k \leq q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}) \\ k\prec j}} q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}) \\ k\succ j}} q_k \ , \forall j \in \mathcal{J}$$

Pero $C_j(\mathcal{I}'(\boldsymbol{x})) = \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} p_k = \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} 1$. Entonces

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} q_j + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} q_k \leq q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} q_k \ , \forall j \in \mathcal{J}$$

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\prec j}} q_j + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} q_k \leq \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} q_k \ , \forall j \in \mathcal{J}$$

Para la instancia $\mathcal{I}'$, la secuencia de las tareas es definida por la regla WSPT, pero como las tareas tienen tiempo de proceso unitario, la secuencia de las tareas se define en orden decreciente según el peso de las tareas. Por lo tanto, se puede sustituir el símbolo $\succ$ por $<$ y el símbolo $\prec$ por $>$. Entonces la condición de optimalidad en $\mathcal{I}'$ es:

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k>j}} q_j + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k<j}} q_k \leq \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k>j}} q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k<j}} q_k \ , \forall j \in \mathcal{J} \tag{4.10}$$

Por otra parte, dado que $\boldsymbol{x}$ es un jump-local de $\mathcal{I}$, se sabe que:

$$C_j(\mathcal{I}(\boldsymbol{x})) + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} q_j \leq q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} q_j \ , \forall j \in \mathcal{J}$$

Pero $C_j(\mathcal{I}(\boldsymbol{x})) = \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} p_k = \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} q_k$. Entonces

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\preceq j}} q_k + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} q_j \leq q_j + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} q_j \ , \forall j \in \mathcal{J}$$

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} q_j \leq \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\prec j}} q_k + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k\succ j}} q_j \ , \forall j \in \mathcal{J}$$

Para la instancia $\mathcal{I}$, la secuencia de las tareas es definida por la regla SPT. Por lo tanto, se puede sustituir el símbolo $\succ$ por $>$ y el símbolo $\prec$ por $<$. Entonces la condición de optimalidad en $\mathcal{I}$ es:

$$\sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k<j}} q_k + \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k>j}} q_j \leq \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k<j}} q_k + \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x})\\ k>j}} q_j \ , \forall j \in \mathcal{J} \tag{4.11}$$

Finalmente, por (4.10) y (4.11), se tiene que la condición de optimalidad es la misma para ambas instancias. Por lo tanto, si $\boldsymbol{x}$ es un jump-óptimo de la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. □

Teorema 4.12 *El factor de aproximación de una solución jump-óptima para el problema* $RP|p_j = 1|\sum w_j C_j$, *se encuentra entre* 1,533 *y* 1,618.

Demostración. Sea $\mathcal{I}$ una instancia del problema $RP||\sum C_j$, $\boldsymbol{x}$ una solución jump-óptima de $\mathcal{I}$, $\mathcal{I}'$ la instancia del problema $RP|p_j = 1|\sum w_j C_j$ obtenida al aplicar la transformada Υ sobre $\mathcal{I}$.
Por el Lema 4.11, si $\boldsymbol{x}$ es un jump-óptimo para la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. Por el Lema 4.10, se tiene que el costo de cualquier solución es el mismo en ambas instancias, $\mathcal{I}$ y $\mathcal{I}'$. Luego, se tiene que para cualquier instancia $\mathcal{I}'$ el factor de aproximación es el mismo que se tiene para la instancia $\mathcal{I}$. Entonces, se tiene que el problema $RP|p_j = 1|\sum w_j C_j$ tiene el mismo factor de aproximación que el problema $RP||\sum C_j$, el cual es establecido por el Teorema 4.7. □

Teorema 4.13 *El factor de aproximación de una solución swap-óptima para el problema* $RP|p_j = 1|\sum w_j C_j$, *se encuentra entre* 1,533 *y* 1,618.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Teorema 4.12 establece una cota superior para el factor de aproximación de un jump-óptimo. Por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 1{,}618$. Por otra parte, la instancia presentada en el Lema 4.6 es un caso particular del problema $RP|p_j = 1|\sum w_j C_j$, y establece una cota inferior para el factor de aproximación. Por lo tanto, $1{,}533 \leq \varphi^S \leq 1{,}618$. □

4.5. Tiempo de completación con tareas unitarias

En notación de scheduling este problema es representado por $RP|p_j = 1|\sum C_j$, donde el peso y el tiempo de proceso de todas las tareas es unitario. Para determinar el factor de aproximación de las soluciones jump-óptimas de este problema se presenta un modelo que permite determinar una cota superior. Luego, este resultado es complementado con el Lema 4.6 para establecer un factor de aproximación ajustado.

Sean $\boldsymbol{x}^*$ y $\boldsymbol{x}$ las soluciones óptima y jump-óptima. Por el Lema 2.6, se tiene que cualquier instancia del problema $R||\sum w_j C_j$ puede ser reducida en términos de elegibilidad de máquina, sin alterar su factor de aproximación. Luego, como el problema $RP|p_j = 1|\sum C_j$ es un caso particular del problema $R||\sum w_j C_j$ es posible asumir que $|\mathcal{M}_j| \leq 2$, para todo $j \in \mathcal{J}$. Luego, para cualquier tarea j, se tiene que $\mathcal{M}_j = \{x_j, x_j^*\}$, es decir las máquinas x_j y x_j^* pueden procesar la tarea j. Si $x_j = x_j^*$, se tiene que la misma máquina procesa la tarea j en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$.

Se define la variable x_{es}, la cual representa la cantidad de máquinas que procesan s y e tareas en las soluciones $\boldsymbol{x}^*$ y $\boldsymbol{x}$ respectivamente. Dado que los tiempos de proceso de las tareas son unitarios, si una máquina procesa k tareas, su aporte al costo total es $T_k = \frac{k(k+1)}{2}$, valor que corresponde al k-ésimo número triangular. Por lo tanto, los costos de las soluciones $\boldsymbol{x}^*$ y $\boldsymbol{x}$ serán:

$$C(\boldsymbol{x}^*) = \sum_{e\geq 0}\sum_{s\geq 0} \frac{s(s+1)}{2} x_{es}$$

$$C(\boldsymbol{x}) = \sum_{e\geq 0}\sum_{s\geq 0} \frac{e(e+1)}{2} x_{es}$$

Con lo anterior, el factor de aproximación de un jump-óptimo se puede escribir como:

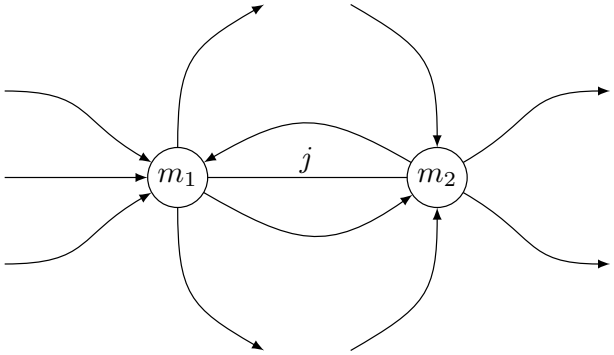
$$\varphi = \frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{\sum_{e\geq 0}\sum_{s\geq 0} e(e+1)x_{es}}{\sum_{e\geq 0}\sum_{s\geq 0} s(s+1)x_{es}} \tag{4.12}$$

Sea m la cantidad de máquinas. En cualquier solución factible, la cantidad de máquinas utilizadas no debe superar a m. Esto es:

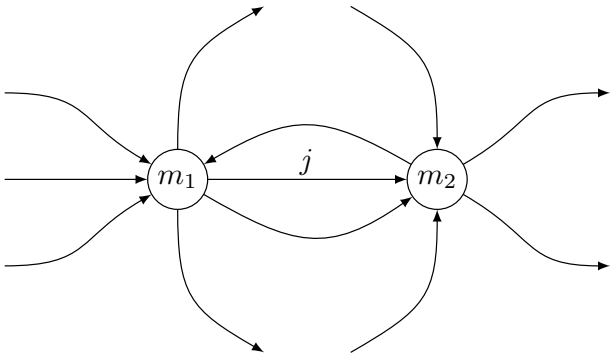
$$\sum_{e\geq 0}\sum_{s\geq 0} x_{es} \leq m \tag{4.13}$$

Además, la cantidad de tareas que se procesan en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$ debe ser la misma. Esto es:

$$\sum_{e\geq 0}\sum_{s\geq 0} e x_{es} = \sum_{e\geq 0}\sum_{s\geq 0} s x_{es} \tag{4.14}$$



(a) El arco j debe ser dirigido.



(b) $x_j = m_2$ y $x_j^* = m_1$

Figura 4.4: Optimalidad local en problema $RP|p_j = 1|\sum C_j$

Para determinar las condiciones de optimalidad local sobre $\boldsymbol{x}$ y $\boldsymbol{x}^*$, se usarán las expresiones (4.4) y (4.5), de tal manera que se cumpla la condición de optimalidad local (4.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}$ en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$. Además, cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen del arco k indica la máquina en la cual es asignada la tarea k en la solución óptima $\boldsymbol{x}^*$. En

cambio, el destino del arco k indica la máquina en la cual es asignada la tarea k en la solución jump-óptima $\boldsymbol{x}$. En la figura 4.4 se presenta un esquema de esta representación. Sea j una tarea con $\mathcal{M}_j = \{m_1, m_2\}$, la máquina m_1 procesa e y s tareas en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$ respectivamente. Mientras que la máquina m_2 procesa i y o tareas en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$ respectivamente. Si $\mathcal{M}_j = \{m_1, m_2\}$ entonces la tarea j debe ser asignada a una de las máquinas en la solución $\boldsymbol{x}$. Entonces el arco j en la figura 4.4a debe ser dirigido.

Luego, si la tarea j es asignada a la máquina m_2 en la solución $\boldsymbol{x}$ (figura 4.4b). Al realizar una movida para esta tarea, se tiene $\delta_j = i$ y $\delta'_j = e + 1$. Luego, para que $\boldsymbol{x}$ sea un óptimo local, al menos se debe cumplir que $i \leq e + 1$. Por otra parte, la solución óptima $\boldsymbol{x}^*$ también debe ser un óptimo local. Si se mueve la tarea j se tiene un ahorro de s y un incremento de $o+1$. Por lo que al menos se debe cumplir que $s \leq o+1$. Con este análisis se puede determinar que la tarea j debe ser representada por un arco dirigido de m_1 a m_2 (figura 4.4b).

Sea y_{es}^{io} la cantidad de arcos dirigidos desde nodos x_{es} a nodos x_{io}. Por las condiciones de optimalidad local, determinadas anteriormente, se tiene que:
$y_{es}^{io} = 0$, para todo $i > e + 1$; y
$y_{es}^{io} = 0$, para todo $s > o + 1$

El modelo propuesto para determinar una cota superior del factor de aproximación usa las siguientes variables:
$\dot{x}_{es}$: fracción de máquinas que procesan e y s tareas en las soluciones $\boldsymbol{x}$ y $\boldsymbol{x}^*$ respectivamente.
$\dot{y}_{es}^{io}$: fracción de arcos dirigidos desde nodos x_{es} a nodos x_{io}.

Notar que de acuerdo a las variables, definidas anteriormente, se tiene que $\dot{x}_{es} = \frac{x_{es}}{m}$ y $\dot{y}_{es}^{io} = \frac{y_{es}^{io}}{m}$.

La expresiones (4.15), (4.16) y (4.17) se obtienen de (4.12), (4.13) y (4.14) respectivamente. (4.18) y (4.19) relacionan la cantidad de nodos y arcos, con la cantidad de máquinas y tareas. La expresión (4.20) representa la condición de optimalidad local para ambas soluciones, óptima y jump-óptima.

$$\text{[WC] maximizar} \qquad \varphi = \frac{\sum_{e\geq 0}\sum_{s\geq 0} e(e+1)\dot{x}_{es}}{\sum_{e\geq 0}\sum_{s\geq 0} s(s+1)\dot{x}_{es}} \tag{4.15}$$

$$\text{sujeto a} \qquad \sum_{e\geq 0}\sum_{s\geq 0} \dot{x}_{es} \leq 1 \tag{4.16}$$

$$\sum_{e\geq 0}\sum_{s\geq 0} e\dot{x}_{es} = \sum_{e\geq 0}\sum_{s\geq 0} s\dot{x}_{es} \tag{4.17}$$

$$\sum_{i\geq 0}\sum_{o\geq 0} \dot{y}^{io}_{es} = s\dot{x}_{es},\ \forall e \geq 0,\ s \geq 0 \tag{4.18}$$

$$\sum_{e\geq 0}\sum_{s\geq 0} \dot{y}^{io}_{es} = i\dot{x}_{io},\ \forall i \geq 0,\ o \geq 0 \tag{4.19}$$

$$\dot{y}^{io}_{es} = 0,\ \forall i > e+1 \vee s > o+1 \tag{4.20}$$

$$\dot{x}_{es} \geq 0,\ \forall e, s \geq 0 \tag{4.21}$$

$$\dot{y}^{io}_{es} \geq 0,\ \forall e, s, i, o \geq 0 \tag{4.22}$$

[WC] es un modelo de optimización fraccionaria, donde φ establece una cota superior para el factor de aproximación de una solución jump-óptima del problema $RP|p_j = 1|\sum C_j$. Para solucionar el modelo se probaron distintas cotas superiores para los índices del modelo, observando una convergencia al valor 1,53337. En la tabla 4.2 se presentan los resultados obtenidos, donde N representa el la cota superior para los índices del modelo. En la figura 4.5 se resume esta información.

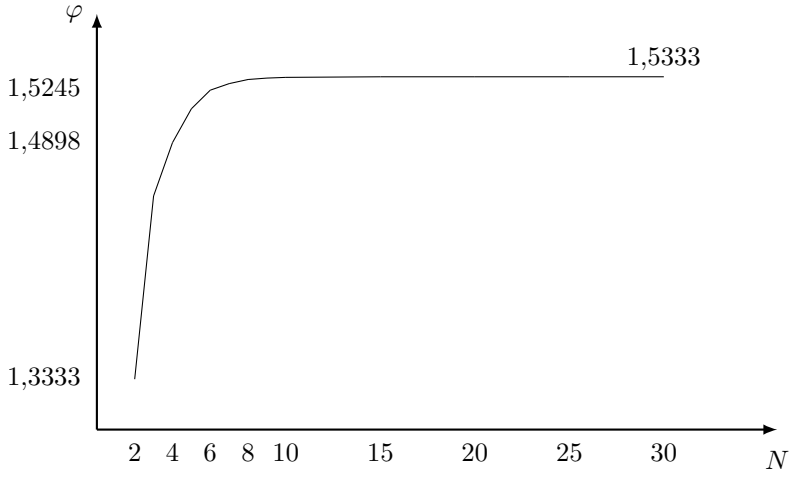


Figura 4.5: Solución del modelo [WC].

A continuación se presenta una conjetura con respecto a la solución del Modelo [WC]. Con esta conjetura, se puede establecer un factor de aproximación ajustado para las soluciones jump-óptimas y swap-óptimas.

Conjetura 4.14 *La solución del modelo [WC] es* $\varphi^* = 1{,}5333$.

Conjetura 4.15 *El factor de aproximación de una solución jump-óptima para el problema* $RP|p_j = 1|\sum C_j$ *es* 1,5333.

N	φ	N	φ
2	1,333333	11	1,533200
3	1,454545	12	1,533288
4	1,489796	13	1,533333
5	1,512195	14	1,533349
6	1,524510	15	1,533357
7	1,528846	20	1,533365
8	1,531564	25	1,533365
9	1,532502	30	1,533367
10	1,532971	35	1,533368
		40	1,533371

Tabla 4.2: Solución del modelo [WC].

DEMOSTRACIÓN. El peor caso para el problema $RP||\sum C_j$ presentado en el Lema 4.6 es una instancia de $RP|p_j = 1|\sum C_j$. Por lo tanto, establece una cota inferior para el factor de aproximación. Con esta cota inferior y la Conjetura 4.14, se tiene que el factor de aproximación de un jump-óptimo para el problema $RP|p_j = 1|\sum C_j$ es 1,5333. □

Conjetura 4.16 *El factor de aproximación de una solución swap-óptima para el problema* $RP|p_j = 1|\sum C_j$ *es* 1,533.

DEMOSTRACIÓN. Sea φ^S el factor de aproximación de un swap-óptimo. La Conjetura 4.14 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 1{,}533$. Por otra parte, el Lema 4.6 establece una cota inferior para el factor de aproximación, ya que la instancia presentada es también un swap-óptimo. Luego, $\varphi^S \geq 1{,}533$. Por lo tanto, $\varphi^S = 1{,}533$. □

Capítulo 5

Máquinas paralelas no-relacionadas

En este capítulo se presentan resultados que permiten determinar la calidad de los óptimos locales para el problema de minimizar el tiempo ponderado de completación en máquinas paralelas no-relacionadas. Se consideran los casos de tiempo de completación ponderado $R||\sum w_jC_j$ y no ponderado $R||\sum C_j$. Las estructuras de vecindario analizadas son jump, swap y exjump.

Para el problema $R||\sum w_jC_j$ se determinó que el factor de aproximación de las soluciones jump-óptimas, swap-óptimas y exjump-óptimas es 2,618. Mientras que para el problema $R||\sum C_j$, el factor de aproximación es 2 para las tres estructuras de vecindario.

5.1. Introducción

Para describir el problema y sus variantes se define la siguiente notación:

$\mathcal{J} = \{1, \ldots, n\}$: el conjunto tareas;

$\mathcal{M} = \{1, \ldots, m\}$: el conjunto de máquinas;

p_{ij}: tiempo necesario para procesar la tarea $j \in \mathcal{J}$ en la máquina $i \in \mathcal{M}$;

w_j: peso o importancia relativa de la tarea j;

$\boldsymbol{z}$: una solución factible del problema;

$C_j(\boldsymbol{z})$: tiempo de completación de la tarea j en la solución $\boldsymbol{z}$;

$C(\boldsymbol{z}) = \sum_{j\in\mathcal{J}} w_jC_j(\boldsymbol{z})$: costo de la solución $\boldsymbol{z}$;

$N_i(\boldsymbol{z})$: conjunto de tareas asignadas a la máquina i en la solución $\boldsymbol{z}$;

$\eta(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} w_jp_{ij}$: suma ponderada de tiempos de proceso en solución $\boldsymbol{z}$.

El problema consiste en determinar una solución $\boldsymbol{z}$, que minimice el tiempo ponderado de completación $C(\boldsymbol{z})$. La solución $\boldsymbol{z}$ establece la máquina a la que cada tarea es asignada y la secuencia en que las tareas son procesadas. Para cada máquina, la secuencia es definida por las reglas WSPT y SPT, para los objetivos $\sum w_j C_j$ y $\sum C_j$ respectivamente (Teoremas 1.1 y 1.2). Adicionalmente, se considera que todas las tareas y máquinas están disponibles desde el inicio, las tareas deben ser procesadas sin interrupción por sólo una máquina y cada máquina puede procesar una tarea a la vez.

Una instancia $\mathcal{I}$ del problema, queda completamente definida por $\mathcal{J}, \mathcal{M}, \boldsymbol{w}, \boldsymbol{P}$. Donde $\boldsymbol{w}$ representa el vector de pesos de las tareas y $\boldsymbol{P}$ la matriz de tiempos de proceso.

Sea $\boldsymbol{z}$ cualquier solución factible de la instancia $\mathcal{I}$. El costo de la solución $\boldsymbol{z}$ puede ser determinado por (5.1), donde el símbolo $k \prec_i j$ indica que la tarea k precede a la tarea j en la máquina i. Por otra parte $k \succ_i j$ indica que la tarea k sucede a la tarea j en la máquina i (los símbolos $\preceq_i$ y $\succeq_i$ además incluyen la tarea j). La precedencia entre las tareas se define por la regla WSPT o SPT según corresponda. En los casos donde exista empate entre dos o más tareas, su precedencia se define en forma arbitraria.

$$C(\boldsymbol{z}) = \eta\left(\boldsymbol{z}\right) + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \prec_i j}} w_j p_{ik} = \eta\left(\boldsymbol{z}\right) + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succ_i j}} w_k p_{ij} \tag{5.1}$$

En notación de scheduling el problema es representado por $R||\sum w_j C_j$. Un caso especial de este problema se obtiene cuando $\boldsymbol{w}$ es constante. Por el Lema 1.3, se tiene que este problema es invariante frente a un re-escalamiento en los pesos de todas las tareas, por lo que se establece su equivalencia con el problema $R||\sum C_j$, donde el peso es unitario para todas las tareas.

En las siguientes secciones se analiza la calidad de los óptimos locales para vecindarios de jump, exjump y swap, para los objetivos de minimización del tiempo de completación ponderado y no ponderado. Para determinar el factor de aproximación de las soluciones localmente óptimas, se determinaron cotas superiores e inferiores para estos factores de aproximación. Para establecer las cotas superiores se utilizó, el mapeo $\varphi : \mathcal{M}^{\mathcal{J}} \to L_2\left([0,\infty)\right)^{\mathcal{M}}$ propuesto por Cole et al. [21], la reducción de elegibilidad de máquina y las desigualdades presentadas en la Sección 2.3. Para determinar las cotas inferiores se presentan instancias que exponen un peor caso para los problemas.

5.2. Tiempo ponderado de completación

Para cualquier instancia $\mathcal{I}$, la asignación óptima de tareas a las máquinas es representada por el vector $\boldsymbol{x}^*$. Donde x_j^* indica la máquina donde es asignada la tarea j, es decir, $x_j^* = i$ si la tarea j es programada en la máquina i. Adicionalmente se define:

$N_i(\boldsymbol{x}^*) = \left\{j \in \mathcal{J} : x_j^* = i\right\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}^*$;

$C_j(\boldsymbol{x}^*)$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}^*$;

$C(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x}^*)$: el costo óptimo;

$\eta(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j p_{x_j^* j}$: suma ponderada de tiempos de proceso en la solución $\boldsymbol{x}^*$.

La solución jump-óptima es representada por el vector $\boldsymbol{x}$. Donde x_j indica la máquina donde es asignada la tarea j, es decir, $x_j = i$ si la tarea j es programada en la máquina i en el óptimo local. Adicionalmente se define:

$N_i(\boldsymbol{x}) = \{j \in \mathcal{J} : x_j = i\}$: el conjunto de tareas asignadas a la máquina i $\boldsymbol{x}$;

$C_j(\boldsymbol{x})$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}$;

$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{x})$: el costo de la solución jump-óptima $\boldsymbol{x}$;

$\eta(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j p_{x_j j}$: suma ponderada de tiempos de proceso en la solución $\boldsymbol{x}$.

Con lo anterior y (5.1), es posible determinar los costos de las soluciones óptima y jump-óptima como:

$$C(\boldsymbol{x}^*) = \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*) \\ k\prec_{x_j^*} j}} w_j p_{x_j^* k} = \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*) \\ k\succ_{x_j^*} j}} w_k p_{x_j^* j} \tag{5.2}$$

$$C(\boldsymbol{x}) = \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x}) \\ k\prec_{x_j} j}} w_j p_{x_j k} = \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x}) \\ k\succ_{x_j} j}} w_k p_{x_j j} \tag{5.3}$$

En la figura 5.1 se presenta un esquema de la movida de jump para la tarea j. Al asignar la tarea j a la máquina t, se produce un cambio en su tiempo de completación, este será la suma entre su nuevo tiempo de proceso (p_{tj}) y el tiempo de proceso de todas las tareas asignadas a la máquina t que preceden a la tarea j. Al retirar la tarea j de la máquina i, el tiempo de completación de todas las tarea que suceden a la tarea j se reduce en p_{ij}. Al asignar la tarea j a la máquina t, el tiempo de completación de todas las tareas que suceden a la tarea j aumenta en p_{tj}.

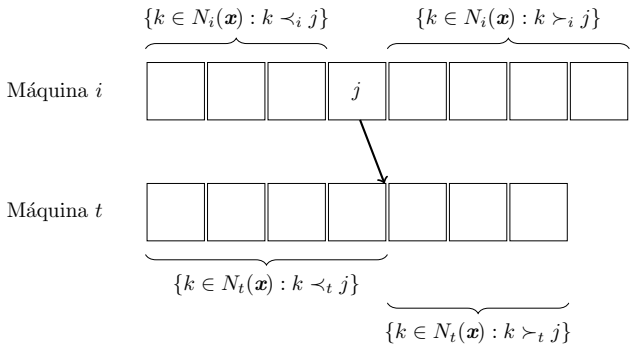


Figura 5.1: Esquema para movida de jump en $R||\sum w_j C_j$

Lema 5.1 *El factor de aproximación de una solución jump-óptima para el problema* $R||\sum w_jC_j$, *es a lo más* $\phi^2 \approx 2{,}618$.

Demostración. Por el Lema 2.6, al aplicar la reducción de elegibilidad de máquina, se tiene que cada tarea $j \in \mathcal{J}$ podrá ser procesada por las máquinas x_j y x_j^*, sin alterar el factor de aproximación. Esto es $\mathcal{M}'_j = \{x_j, x_j^*\}$ para todo $j \in \mathcal{J}$. Sea δ_j el ahorro en $C(\boldsymbol{x})$ que se obtiene al retirar la tarea j de la máquina x_j; y sea δ'_j el incremento en $C(\boldsymbol{x})$ al asignar la tarea j a la máquina x_j^*.

$$\delta_j = w_jC_j(\boldsymbol{x}) + p_{x_jj} \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ_{x_j} j}} w_k \tag{5.4}$$

$$\delta'_j = w_jp_{x_j^*j} + w_j \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^*k} + p_{x_j^*j} \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k \tag{5.5}$$

Para que $\boldsymbol{x}$ sea un jump-óptima, se debe cumplir la condición $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}$. Por lo tanto, la condición de optimalidad local para el vecindario de jump es:

$$w_jC_j(\boldsymbol{x}) + p_{x_jj} \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ_{x_j} j}} w_k \leq w_jp_{x_j^*j} + w_j \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^*k} + p_{x_j^*j} \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k \tag{5.6}$$

Sumando $w_jp_{x_jj}$ a ambos lados

$$w_jC_j(\boldsymbol{x}) + p_{x_jj} \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succeq_{x_j} j}} w_k \leq w_jp_{x_jj} + w_jp_{x_j^*j} + w_j \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec_{x_j^*} j}} p_{x_j^*k} + p_{x_j^*j} \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ_{x_j^*} j}} w_k$$

Agrupando las últimas dos componentes y considerando que: si $k \prec_{x_j^*} j$, entonces $\frac{p_{x_j^*k}}{w_k} \leq \frac{p_{x_j^*j}}{w_j}$; si $k \succ_{x_j^*} j$, entonces $\frac{p_{x_j^*j}}{w_j} \leq \frac{p_{x_j^*k}}{w_k}$.

$$w_jC_j(\boldsymbol{x}) + p_{x_jj} \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succeq_{x_j} j}} w_k \leq w_jp_{x_jj} + w_jp_{x_j^*j} + \sum_{k \in N_{x_j^*}(\boldsymbol{x})} w_jw_k \min\left\{\frac{p_{x_j^*k}}{w_k}, \frac{p_{x_j^*j}}{w_j}\right\}$$

Sumando para todo $j \in \mathcal{J}$, o bien, $i \in \mathcal{M}$ y $j \in N_i(\boldsymbol{x}^*)$, y usando $\eta(\boldsymbol{x}^*)$ y $\eta(\boldsymbol{x})$

$$\sum_{j \in \mathcal{J}} w_jC_j(\boldsymbol{x}) + \sum_{j \in \mathcal{J}} \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succeq_{x_j} j}} w_kp_{x_jj} \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{x}^*)} \sum_{k \in N_i(\boldsymbol{x})} w_jw_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}$$

Usando (5.3) se tiene

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{x}^*)} \sum_{k \in N_i(\boldsymbol{x})} w_jw_k \min\left\{\frac{p_{ik}}{w_k}, \frac{p_{ij}}{w_j}\right\}$$

Por (2.5) se tiene

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \int_0^\infty f_i(y) f_i^*(y) dy \tag{5.7}$$

Aplicando la desigualdad del Lema 2.9

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i \in \mathcal{M}} \int_0^\infty \left(\frac{\beta}{2} f_i^{*2}(y) + \frac{1}{2\beta} f_i^2(y) \right) dy$$

Ordenando y usando (2.4)

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \frac{\beta}{2} \left(2C(\boldsymbol{x}^*) - \eta(\boldsymbol{x}^*) \right) + \frac{1}{2\beta} \left(2C(\boldsymbol{x}) - \eta(\boldsymbol{x}) \right)$$

$$\left(2 - \frac{1}{\beta} \right) C(\boldsymbol{x}) \leq \beta C(\boldsymbol{x}^*) + \left(1 - \frac{1}{2\beta} \right) \eta(\boldsymbol{x}) + \left(1 - \frac{\beta}{2} \right) \eta(\boldsymbol{x}^*)$$

Suponiendo que:

$$1 - \frac{1}{2\beta} \geq 0 \tag{5.8}$$

$$1 - \frac{\beta}{2} \geq 0 \tag{5.9}$$

De (5.2) y (5.3), se tiene que $\eta(\boldsymbol{x}^*) \leq C(\boldsymbol{x}^*)$ y $\eta(\boldsymbol{x}) \leq C(\boldsymbol{x})$. Entonces:

$$\left(1 - \frac{1}{2\beta} \right) C(\boldsymbol{x}) \leq C(\boldsymbol{x}^*) \left(1 + \frac{\beta}{2} \right)$$

Haciendo $\beta = \phi = \frac{1+\sqrt{5}}{2} \approx 1{,}618$ (número áureo), se cumple (5.8) y (5.9), y se alcanza el mayor valor para $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)}$.

$$\left(\frac{2\phi - 1}{2\phi} \right) C(\boldsymbol{x}) \leq \left(\frac{2 + \phi}{2} \right) (\boldsymbol{x}^*)$$

Usando $1 + \phi = \phi^2$, se tiene:

$$C(\boldsymbol{x}) \leq \left(\frac{2\phi + \phi^2}{2\phi - 1} \right) C(\boldsymbol{x}^*) = \left(\frac{3\phi + 1}{2\phi - 1} \right) C(\boldsymbol{x}^*) = \phi^2 C(\boldsymbol{x}^*)$$

□

A continuación, se presenta una instancia del problema $R||\sum w_j C_j$, que expone un peor caso para los vecindarios de jump, exjump y swap.

Lema 5.2 *El factor de aproximación de una solución jump-óptima, exjump-óptima o swap-óptima, para el problema $R||\sum w_j C_j$, es al menos $\phi^2 \approx 2{,}618$.*

Demostración. Sea $\mathcal{I}$ una instancia con $m = n + 1$ máquinas y dos conjuntos de tareas, $\mathcal{J}_1$ y $\mathcal{J}_2$. El conjunto $\mathcal{J}_1$ tiene n tareas, el peso de la tarea $j \in \mathcal{J}_1$ es $w_j = \phi^{2-j}$, $j = 1, \ldots, n$; con $\phi = \frac{1+\sqrt{5}}{2}$, el número áureo. Cada tarea $j = 1, \ldots, n$ puede ser asignada en las máquinas

j y $j+1$. Para las otras máquinas el tiempo de proceso es ∞. El tiempo de proceso de las tareas del conjunto $\mathcal{J}_1$ es definido por:

$$p_{ij} = \begin{cases} \phi^j & , \ \forall i = j+1, \ j \geq 1 \\ \phi^{j-2} & , \ \forall i = j, \ j \geq 2 \\ \phi & , \ \text{para } (i,j) = (1,1) \\ \infty & , \ \text{en otro caso} \end{cases}$$

El conjunto $\mathcal{J}_2$ tiene $n+1$ tareas, el peso de cada tarea en este conjunto es unitario. Cada tarea $j = 1, \ldots, n+1$ puede ser asignada únicamente a la máquina j. El tiempo de proceso de las tareas del conjunto $\mathcal{J}_2$ es definido por:

$$\xi_{ij} = \begin{cases} \xi_j & , \ \forall i = j, \ j \geq 1 \\ \infty & , \ \text{en otro caso} \end{cases}$$

donde $\xi_1 > \xi_2 > \ldots > \xi_n > \xi_{n+1}$, con $\xi_j \to 0, \ \forall j = 1, \ldots, n+1$.
En cualquier solución factible, la j-ésima tarea del conjunto $\mathcal{J}_2$ es asignada a la máquina j. En la solución óptima $\boldsymbol{x}^*$, la j-ésima tarea del conjunto $\mathcal{J}_1$ es asignada a la máquina j. Mientras que en la solución jump-óptima $\boldsymbol{x}$, es asignada a la máquina $j+1$.

En la figura 5.2 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen de cada arco, indica la máquina en la cual es asignada la tarea en la solución óptima $\boldsymbol{x}^*$. El destino de cada arco, indica la máquina en la cual es asignada la tarea en la solución local $\boldsymbol{x}$. Las tareas del conjunto $\mathcal{J}_2$ siempre serán secuenciadas primero, ya que su cociente de Smith (CS) es $\frac{1}{\xi_j} \to \infty, \ \forall j \in \mathcal{J}_2$. Mientras que el CS de las tareas del conjunto $\mathcal{J}_1$, es a lo más 1.

En la figura 5.3 se presentan las movidas factibles para la solución jump-óptima. En la solución $\boldsymbol{x}$, sólo las tareas $j \in \mathcal{J}_1$ pueden ser movidas. Para verificar que $\boldsymbol{x}$ es un óptimo local de $\mathcal{I}$, se usarán las expresiones (5.4) y (5.5), de tal manera que se cumpla la condición de optimalidad local (5.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}_1$.

Para $k > 2$ (figura 5.3a), al mover la tarea k de la máquina $k+1$ a la máquina k, se genera un ahorro $\delta_k = w_k(\xi_{k+1} + p_{k+1,k}) = w_k(\xi_{k+1} + \phi^k)$. El CS de la tarea k en la máquina k es $r_{kk} = \frac{w_k}{p_{kk}} = \frac{\phi^{2-k}}{\phi^{k-2}} = \phi^{4-2k}$, y el CS de la tarea $k-1$ en la máquina k es $r_{k,k-1} = \frac{w_{k-1}}{p_{k,k-1}} = \frac{\phi^{3-k}}{\phi^{k-1}} = \phi^{4-2k}$. Como ambas tareas tienen el mismo CS, pueden ser secuenciadas en cualquier orden. El incremento en $C(\boldsymbol{x})$ es $\delta'_k = w_k\left(\xi_k + p_{k,k-1} + p_{kk}\right) = w_k\left(\xi_k + \phi^{k-1} + \phi^{k-2}\right) = w_k\left(\xi_k + \phi^k\right)$. Como $\xi_{k+1} < \xi_k$, se tiene que $\delta_k < \delta'_k$. Luego, mover la tarea k a la máquina k, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea k, $k > 2$.
Por otra parte, mover la tarea 2 de la máquina 3 a la máquina 2 (figura 5.3b), genera un ahorro $\delta_2 = w_2(\xi_3 + p_{32}) = w_2(\xi_3 + \phi^2)$. El CS de la tarea 2 en la máquina 2 es $r_{22} = \frac{w_2}{p_{22}} = \frac{\phi^0}{\phi^0} = 1$, y el CS de la tarea 1 en la máquina 2 es $r_{21} = \frac{w_1}{p_{21}} = \frac{\phi}{\phi} = 1$. Como ambas tareas tienen el mismo CS, pueden ser secuenciadas en cualquier orden. Luego, se tiene un incremento $\delta'_2 = w_2\left(\xi_2 + p_{21} + p_{22}\right) = w_2\left(\xi_2 + \phi + 1\right) = w_2\left(\xi_2 + \phi^2\right)$. Como $\xi_3 < \xi_2$, se tiene que $\delta_2 < \delta'_2$. Luego, mover la tarea 2 a la máquina 2, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea 2.
Finalmente, mover la tarea 1 de la máquina 2 a la máquina 1 (figura 5.3c), genera un ahorro

$\delta_1 = w_1(\xi_2 + p_{21}) = w_1(\xi_2 + \phi)$. El incremento al asignar la tarea 1 a la máquina 1 es $\delta_1' = w_1(\xi_1 + p_{11}) = w_1(\xi_1 + \phi)$. Como $\xi_2 < \xi_1$, se tiene que $\delta_1 < \delta_1'$. Luego, mover la tarea 1 a la máquina 1, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea 1. De manera similar, se puede verificar que el óptimo global es también un óptimo local.

Sea $\Xi = \sum_{j\in\mathcal{J}_2} \xi_j$; $\Pi_1 = \sum_{j=1}^{n} w_j\xi_j$; $\Pi_2 = \sum_{j=1}^{n} w_j\xi_{j+1}$.
El costo de las soluciones óptima y jump-óptima es[1]:

$$\begin{aligned} C(\boldsymbol{x}^*) &= \sum_{j\in\mathcal{J}_2} \xi_j + \sum_{j=1}^{n} w_j\left(\xi_j + p_{jj}\right) \\ &= \Xi + \Pi_1 + \sum_{j=1}^{n} w_j p_{jj} \\ &= \Xi + \Pi_1 + w_1 p_{11} + \sum_{j=2}^{n} w_j\left(\xi_j + p_{jj}\right) \\ &= \Xi + \Pi_1 + \phi^2 + \sum_{j=2}^{n} \phi^{2-j}\phi^{j-2} \\ &= \Xi + \Pi_1 + \phi^2 + n - 1 \\ &= \Xi + \Pi_1 + \phi + n \end{aligned}$$

$$\begin{aligned} C(\boldsymbol{x}) &= \sum_{j\in\mathcal{J}_2} \xi_j + \sum_{j=1}^{n} w_j\left(\xi_{j+1} + p_{j+1,j}\right) \\ &= \Xi + \Pi_2 + \sum_{j=1}^{n} w_j p_{j+1,j} = \Xi + \Pi_2 + \sum_{j=1}^{n} \phi^{2-j}\phi^{j} \\ &= \Xi + \Pi_2 + n\phi^2 \end{aligned}$$

Luego, el factor de aproximación de un jump-óptimo es:

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{n\phi^2 + \Xi + \Pi_2}{n + \phi + \Xi + \Pi_1} = \frac{\phi^2 + \frac{\Xi+\Pi_2}{n}}{1 + \frac{\phi+\Xi+\Pi_1}{n}}$$

Para n suficientemente grande y cualquier $\varepsilon > 0$, se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \phi^2 - \varepsilon$.

Como todas las movidas posibles en la solución $\boldsymbol{x}$, generan un deterioro en el costo $C(\boldsymbol{x})$, la solución $\boldsymbol{x}$ es también un exjump-óptimo.

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}_1$ tal que $j \neq k$, se tiene $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo.

□

[1] se usa la igualdad $\phi^2 = 1 + \phi$

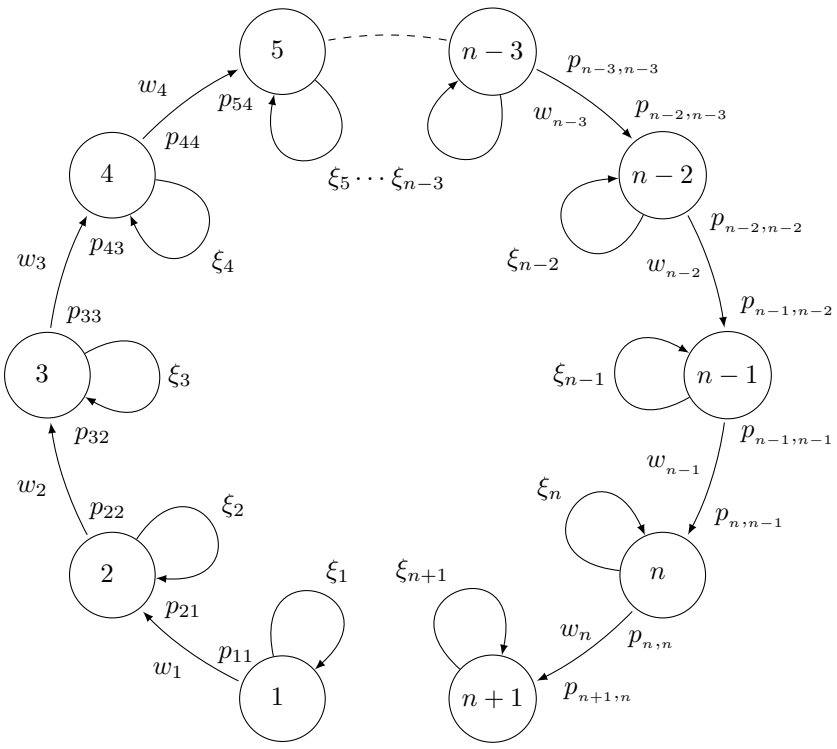


Figura 5.2: Instancia peor caso $R||\sum w_jC_j$

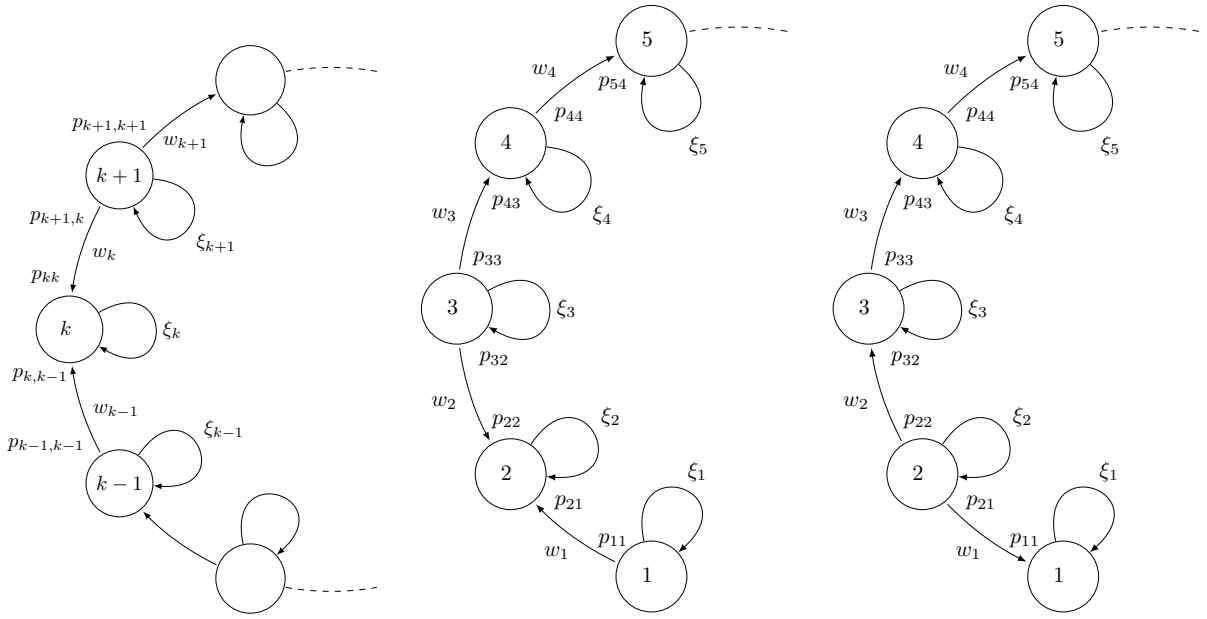
Con los Lemas 5.1, 5.2, se puede determinar un factor de aproximación ajustado para las soluciones jump-óptima, exjump-óptima y swap-óptima; Teoremas 5.3, 5.4 y 5.5 respectivamente.

Teorema 5.3 *El factor de aproximación de una solución jump-óptima para el problema* $R||\sum w_jC_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación. Luego, $\varphi^J \leq \phi^2$. Por otra parte, el Lema 5.2 establece una cota inferior para el factor de aproximación. Luego, $\varphi^J \geq \phi^2$. Por lo tanto, $\varphi^J = \phi^2$. □

Teorema 5.4 *El factor de aproximación de una solución exjump-óptima para el problema* $R||\sum w_jC_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^L el factor de aproximación de un exjump-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, toda solución exjump-óptima es también un jump-óptimo. Luego, $\varphi^L \leq \phi^2$. Por otra parte, el Lema 5.2 establece una cota inferior para el factor de aproximación. Luego, $\varphi^L \geq \phi^2$. Por lo tanto, $\varphi^L = \phi^2$. □



(a) $k > 2$ (b) $k = 2$ (c) $k = 1$

Figura 5.3: Movidas factibles para peor caso $R||\sum w_j C_j$

Teorema 5.5 *El factor de aproximación de una solución swap-óptima para el problema* $R||\sum w_j C_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq \phi^2$. Por otra parte, el Lema 5.2 establece una cota inferior para el factor de aproximación. Luego, $\varphi^S \geq \phi^2$. Por lo tanto, $\varphi^S = \phi^2$. □

5.3. Tiempo de completación

En notación de scheduling este problema es representado por $R||\sum C_j$, donde el peso de todas las tareas es unitario. Por el Lema 1.3, se tiene que este problema es equivalente al caso donde el peso es constante para todas las tareas, $w_j = w$ para todo $j \in \mathcal{J}$.

Dado que el peso de cada tarea es unitario, se tiene:
$\eta(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} p_{x_j^* j}$
$\eta(\boldsymbol{x}\ ) = \sum_{j\in\mathcal{J}} p_{x_j j}$

Para este problema, el mapeo $\varphi : \mathcal{M}^{\mathcal{J}} \to L_2\left([0,\infty)\right)^{\mathcal{M}}$ propuesto por Cole et al. [21], es ligeramente distinto. De (2.2), se tiene que:

$$f_i(y) = \sum_{j\in N_i(\boldsymbol{z}):p_{ij}\geq y} 1$$

Como $f_i(y)$ es un contador de tareas asignadas a la máquina i con tiempo de proceso superior o igual a y, se puede imponer condiciones de integralidad sobre los elementos $f_i(y)$.

Para determinar el factor de aproximación de los óptimos locales para los vecindarios de jump, exjump y swap, se determinará una cota superior para el factor de aproximación de las soluciones jump-óptimas. Luego, se presentará una instancia (peor caso) para determinar una cota inferior para los vecindarios de jump, exjump y swap.

Lema 5.6 *El factor de aproximación de una solución jump-óptima para el problema* $R||\sum C_j$, *es a lo más* 2.

Demostración. De (5.7) se tiene

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i\in\mathcal{M}} \int_0^\infty f_i(y) f_i^*(y)\, dy$$

Por la integralidad de los $f_i(y)$ y $f_i^*(y)$, se puede usar la desigualdad del Lema 2.7. Entonces:

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \sum_{i\in\mathcal{M}} \int_0^\infty \left(\frac{f_i^2(y)}{2} + \frac{f_i^{*2}(y)}{2} - \frac{f_i(y)}{2} + \frac{f_i^*(y)}{2} \right) dy$$

Para este problema, se tiene que $w_j = 1$, para todo $j \in \mathcal{J}$. Por (2.3), se tiene que:
$\sum_{i\in\mathcal{M}} \int_0^\infty f_i(y)\, dy = \eta(\boldsymbol{x})$; $\sum_{i\in\mathcal{M}} \int_0^\infty f_i^*(y)\, dy = \eta(\boldsymbol{x}^*)$
Ordenando y usando (2.4) se tiene:

$$2C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}) + \eta(\boldsymbol{x}^*) + \frac{2C(\boldsymbol{x}) - \eta(\boldsymbol{x})}{2} + \frac{2C(\boldsymbol{x}^*) - \eta(\boldsymbol{x}^*)}{2} - \frac{\eta(\boldsymbol{x})}{2} + \frac{\eta(\boldsymbol{x}^*)}{2}$$

Agrupando

$$C(\boldsymbol{x}) \leq \eta(\boldsymbol{x}^*) + C(\boldsymbol{x}^*)$$

Por (5.2), se tiene que $\eta(\boldsymbol{x}^*) \leq C(\boldsymbol{x}^*)$. Finalmente se tiene $C(\boldsymbol{x}) \leq 2C(\boldsymbol{x}^*)$. □

A continuación, se presenta una instancia del problema $R||\sum C_j$, que expone un peor caso para los vecindarios de jump, exjump y swap.

Lema 5.7 *El factor de aproximación de una solución jump-óptima, exjump-óptima o swap-óptima óptima, para el problema* $R||\sum C_j$, *es al menos* 2.

Demostración. Sea $\mathcal{I}$ una instancia con $m = n+1$ máquinas y dos conjuntos de tareas, $\mathcal{J}_1$ y $\mathcal{J}_2$. El conjunto $\mathcal{J}_1$ tiene n tareas, cada tarea $j = 1, \ldots, n$ puede ser asignada en las máquinas j y $j+1$. Para las otras máquinas el tiempo de proceso es ∞. El tiempo de proceso de las tareas del conjunto $\mathcal{J}_1$ es definido por:

$$p_{ij} = \begin{cases} 2^{j-1} & , \ \forall i = j+1, \ j \geq 1 \\ 2^{j-2} & , \ \forall i = j, \ j \geq 2 \\ 1 & , \ \text{para } (i,j) = (1,1) \\ \infty & , \ \text{en otro caso} \end{cases}$$

El conjunto $\mathcal{J}_2$ tiene $n+1$ tareas, cada tarea $j = 1, \ldots, n+1$ puede ser asignada únicamente a la máquina j. El tiempo de proceso de las tareas del conjunto $\mathcal{J}_2$ es definido por:

$$\xi_{ij} = \begin{cases} \xi_j & , \ \forall i = j, \ j \geq 1 \\ \infty & , \text{ en otro caso} \end{cases}$$

donde $\xi_1 > \xi_2 > \ldots > \xi_n > \xi_{n+1}$, con $\xi_j \to 0, \ \forall j = 1, \ldots, n+1$.
En cualquier solución factible, la j-ésima tarea del conjunto $\mathcal{J}_2$ es asignada a la máquina j. En la solución óptima $\boldsymbol{x}^*$, la j-ésima tarea del conjunto $\mathcal{J}_1$ es asignada a la máquina j. Mientras que en la solución jump-óptima $\boldsymbol{x}$, es asignada a la máquina $j+1$.

En la figura 5.2 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen de cada arco, indica la máquina en la cual es asignada la tarea en la solución óptima $\boldsymbol{x}^*$. El destino de cada arco, indica la máquina en la cual es asignada la tarea en la solución local $\boldsymbol{x}$. Las tareas del conjunto $\mathcal{J}_2$ siempre serán secuenciadas primero, ya que su tiempo de proceso es $\xi_j \to 0, \ \forall j \in \mathcal{J}_2$. Mientras que el tiempo de proceso de las tareas del conjunto $\mathcal{J}_1$, es al menos 1.

En la figura 5.3 se presentan las movidas factibles para la solución jump-óptima. En la solución $\boldsymbol{x}$, sólo las tareas $j \in \mathcal{J}_1$ pueden ser movidas. Para verificar que $\boldsymbol{x}$ es un óptimo local de $\mathcal{I}$, se usarán las expresiones (5.4) y (5.5), de tal manera que se cumpla la condición de optimalidad local (5.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}_1$.

Para $k > 2$ (figura 5.3a), al mover la tarea k de la máquina $k+1$ a la máquina k, se genera un ahorro $\delta_k = \xi_{k+1} + p_{k+1,k} = \xi_{k+1} + 2^{k-1}$. El tiempo de proceso de la tarea k en la máquina k es idéntico al tiempo de proceso de la tarea $k-1$ en la máquina k. Como ambas tareas tienen el mismo tiempo de proceso, pueden ser secuenciadas en cualquier orden. El incremento en $C(\boldsymbol{x})$ es $\delta'_k = \xi_k + p_{k,k-1} + p_{kk} = \xi_k + 2^{k-2} + 2^{k-2} = \xi_k + 2^{k-1}$. Como $\xi_{k+1} < \xi_k$, se tiene que $\delta_k < \delta'_k$. Luego, mover la tarea k a la máquina k, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea k, $k > 2$.
Por otra parte, mover la tarea 2 de la máquina 3 a la máquina 2 (figura 5.3b), genera un ahorro $\delta_2 = \xi_3 + p_{32} = \xi_3 + 2$. El tiempo de proceso de la tarea 2 en la máquina 2 es idéntico al tiempo de proceso de la tarea 1 en la máquina 2. Como ambas tareas tienen el mismo tiempo de proceso, pueden ser secuenciadas en cualquier orden. Luego, se tiene un incremento $\delta'_2 = \xi_2 + p_{21} + p_{22} = \xi_2 + 1 + 1 = \xi_2 + 2$. Como $\xi_3 < \xi_2$, se tiene que $\delta_2 < \delta'_2$. Luego, mover la tarea 2 a la máquina 2, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea 2.
Finalmente, mover la tarea 1 de la máquina 2 a la máquina 1 (figura 5.3c), genera un ahorro $\delta_1 = \xi_2 + p_{21} = \xi_2 + 1$. El incremento al asignar la tarea 1 a la máquina 1 es $\delta'_1 = \xi_1 + p_{11} = \xi_1 + 1$. Como $\xi_2 < \xi_1$, se tiene que $\delta_1 < \delta'_1$. Luego, mover la tarea 1 a la máquina 1, empeora el costo $C(\boldsymbol{x})$. Por lo tanto, no se realizan movidas para la tarea 1.
De manera similar, se puede verificar que el óptimo global es también un óptimo local.

Sea $\Xi = \sum_{j \in \mathcal{J}_2} \xi_j$. Luego, el costo de las soluciones óptima y jump-óptima es:

$$\begin{aligned} C(\boldsymbol{x}^*) &= \sum_{j \in \mathcal{J}_2} \xi_j + \sum_{j=1}^{n} (\xi_j + p_{jj}) = \Xi + \Xi - \xi_{n+1} + \sum_{j=1}^{n} p_{jj} \\ &= 2\Xi - \xi_{n+1} + p_{11} + \sum_{j=2}^{n} p_{jj} = 2\Xi - \xi_{n+1} + 1 + \sum_{j=2}^{n} 2^{j-2} \\ &= 2\Xi - \xi_{n+1} + 1 + \sum_{j=0}^{n-2} 2^j = 2\Xi - \xi_{n+1} + 2^{n-1} \end{aligned}$$

$$\begin{aligned} C(\boldsymbol{x}) &= \sum_{j \in \mathcal{J}_2} \xi_j + \sum_{j=1}^{n} (\xi_{j+1} + p_{j+1,j}) = \Xi + \Xi - \xi_1 + \sum_{j=1}^{n} p_{j+1,j} \\ &= 2\Xi - \xi_1 + \sum_{j=1}^{n} 2^{j-1} = 2\Xi - \xi_1 + \sum_{j=0}^{n-1} 2^j \\ &= 2\Xi - \xi_1 + 2^n - 1 \end{aligned}$$

Luego, el factor de aproximación de un jump-óptimo es:

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{2^n + 2\Xi - \xi_1 - 1}{2^{n-1} + 2\Xi - \xi_{n+1}} = \frac{2 + \frac{2\Xi - \xi_1 - 1}{2^{n-1}}}{1 + \frac{2\Xi - \xi_{n+1}}{2^{n-1}}}$$

Para n suficientemente grande y cualquier $\varepsilon > 0$, se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = 2 - \varepsilon$.

Como todas las movidas posibles en la solución $\boldsymbol{x}$, generan un deterioro en el costo $C(\boldsymbol{x})$, la solución $\boldsymbol{x}$ es también un exjump-óptimo.

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}_1$ tal que $j \neq k$, se tiene que $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo. □

Con los Lemas 5.6, 5.7 y otras consideraciones, se puede determinar un factor de aproximación ajustado para las soluciones jump-óptima, exjump-óptima y swap-óptima; Teoremas 5.8, 5.9 y 5.10 respectivamente.

Teorema 5.8 *El factor de aproximación de una solución jump-óptima para el problema* $R||\sum C_j$ *es* 2.

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación. Luego, $\varphi^J \leq 2$. Por otra parte, el Lema 5.7 establece una cota inferior para el factor de aproximación. Luego, $\varphi^J \geq 2$. Por lo tanto, $\varphi^J = 2$. □

Teorema 5.9 *El factor de aproximación de una solución exjump-óptima para el problema* $R||\sum C_j$ *es* 2.

Demostración. Sea φ^L el factor de aproximación de un exjump-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, toda solución exjump-óptima es también un jump-óptimo. Luego, $\varphi^L \leq 2$. Por otra parte, el Lema 5.7 establece una cota inferior para el factor de aproximación. Luego, $\varphi^L \geq 2$. Por lo tanto, $\varphi^L = 2$. □

Teorema 5.10 *El factor de aproximación de una solución swap-óptima para el problema* $R||\sum C_j$ *es* 2.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 2$. Por otra parte, el Lema 5.7 establece una cota inferior para el factor de aproximación. Luego, $\varphi^S \geq 2$. Por lo tanto, $\varphi^S = 2$. □

Capítulo 6

Máquinas paralelas uniformes restringidas

En este capítulo se presentan resultados que permiten determinar la calidad de los óptimos locales para el problema de minimizar el tiempo ponderado de completación en máquinas paralelas uniformes restringidas $RQ||\sum w_jC_j$. Además del caso general, se consideran los casos: no ponderado $RQ||\sum C_j$, tiempo de proceso constante ponderado $RQ|p_j = 1|\sum w_jC_j$ y no ponderado $RQ|p_j = 1|\sum C_j$. Las estructuras de vecindario analizadas son jump, exjump y swap.

Para el problema $RQ||\sum w_jC_j$ se determinó que el factor de aproximación de las soluciones jump-óptimas, exjump-óptimas y swap-óptimas es 2,618. Para las variantes del problema, el factor de aproximación para estos vecindarios es 2.

6.1. Introducción

Para describir el problema y sus variantes se define la siguiente notación:

$\mathcal{J} = \{1, \ldots, n\}$: el conjunto tareas;

$\mathcal{M} = \{1, \ldots, m\}$: el conjunto de máquinas;

$\mathcal{M}_j \subseteq \mathcal{M}$: conjunto de máquinas que puede procesar la tarea j;

p_j: tiempo de proceso requerido por la tarea j;

w_j: peso o importancia relativa de la tarea j;

s_i: velocidad de la máquina i;

$\boldsymbol{z}$: una solución factible del problema;

$C_j(\boldsymbol{z})$: tiempo de completación de la tarea j en la solución $\boldsymbol{z}$;

$C(\boldsymbol{z}) = \sum_{j\in\mathcal{J}} w_j C_j(\boldsymbol{z})$: costo de la solución $\boldsymbol{z}$;

$N_i(\boldsymbol{z})$: conjunto de tareas asignadas a la máquina i en la solución $\boldsymbol{z}$;

$\eta(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} w_j \frac{p_j}{s_i}$: suma ponderada de tiempos de proceso en la solución $\boldsymbol{z}$.

En este ambiente de máquinas, cada tarea $j \in \mathcal{J}$ puede ser procesada, sin interrupción, por un conjunto restringido de máquinas $\mathcal{M}_j \subseteq \mathcal{M}$. Cada tarea $j \in \mathcal{J}$ requiere p_j unidades de tiempo de proceso, el cual debe ser realizado por alguna máquina $i \in \mathcal{M}_j$, que opera a una velocidad s_i. Por lo tanto, el tiempo requerido para procesar la tarea j en la máquina i es $\frac{p_j}{s_i}$. Cada máquina puede procesar una tarea a la vez, y todas las tareas y máquinas están disponibles desde el inicio. El problema consiste en determinar una solución $\boldsymbol{z}$, que minimice $C(\boldsymbol{z})$. La solución $\boldsymbol{z}$ establece la máquina a la que cada tarea es asignada y la secuencia en que las tareas son procesadas. Para cada máquina, la secuencia es definida por las reglas WSPT y SPT, para los objetivos $\sum w_j C_j$ y $\sum C_j$ respectivamente (Teoremas 1.1 y 1.2).

Una instancia $\mathcal{I}$ del problema, queda completamente definida por $\mathcal{M}_j$, $\mathcal{J}$, $\mathcal{M}$, $\boldsymbol{w}$, $\boldsymbol{p}$ y $\boldsymbol{s}$. Donde $\boldsymbol{w}$ y $\boldsymbol{p}$ son los vectores de pesos y tiempos de proceso de las tareas respectivamente; $\boldsymbol{s}$ es el vector de velocidad de las máquinas.

Sea $\boldsymbol{z}$ cualquier solución factible de la instancia $\mathcal{I}$. El costo de la solución $\boldsymbol{z}$ puede ser determinado por (6.1), donde el símbolo $k \prec j$ indica que la tarea k precede a la tarea j. Por otra parte $k \succ j$ indica que la tarea k sucede a la tarea j (los símbolos $\preceq$ y $\succeq$ además incluyen la tarea j). La precedencia entre las tareas se define por la regla WSPT o SPT según corresponda. En los casos donde exista empate entre dos o más tareas, su precedencia se define en forma arbitraria.

$$C(\boldsymbol{z}) = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\preceq j}} w_j \frac{p_k}{s_i} = \sum_{i\in\mathcal{M}} \sum_{j\in N_i(\boldsymbol{z})} \sum_{\substack{k\in N_i(\boldsymbol{z}) \\ k\succeq j}} w_k \frac{p_j}{s_i} \tag{6.1}$$

En notación de scheduling el problema es representado por $RQ||\sum w_j C_j$ o $Q|\mathcal{M}_j|\sum w_j C_j$. A partir de este problema general, es posible obtener algunos casos particulares (las equivalencias quedan demostradas por el Lema 1.3):

1. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son arbitrarios, se tiene el problema $RQ||\sum w_j C_j$
2. Si $\boldsymbol{w}$ es constante, el problema resultante es equivalente a $RQ||\sum C_j$
3. Si $\boldsymbol{p}$ es constante, el problema resultante es equivalente a $RQ|p_j = 1|\sum w_j C_j$
4. Si $\boldsymbol{w}$ y $\boldsymbol{p}$ son constantes, el problema resultante es equivalente a $RQ|p_j = 1|\sum C_j$

El problema estudiado en este capítulo es un caso particular del problema $R||\sum w_j C_j$. Por lo tanto, las cotas superiores presentadas en el Capítulo 5 son válidas para cada caso particular.

En las siguientes secciones se analiza la calidad de los óptimos locales para vecindarios de jump, exjump y swap para los cuatro casos. Para determinar el factor de aproximación de los óptimos locales, se establecen cotas inferiores para estos factores de aproximación. Las cuales fueron complementadas con las cotas superiores establecidas para los problemas $R||\sum w_jC_j$ y $R||\sum C_j$ (Capítulo 5). Para determinar las cotas inferiores se presentan instancias que exponen un peor caso para los problemas.

6.2. Tiempo ponderado de completación

Para cualquier instancia $\mathcal{I}$, la asignación óptima de tareas a las máquinas es representada por el vector $\boldsymbol{x}^*$. Donde x_j^* indica la máquina donde es asignada la tarea j, es decir, $x_j^* = i$ si la tarea j es programada en la máquina i. Adicionalmente se define:

$N_i(\boldsymbol{x}^*) = \{j \in \mathcal{J} : x_j^* = i\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}^*$;

$C_j(\boldsymbol{x}^*)$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}^*$;

$C(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_jC_j(\boldsymbol{x}^*)$: el costo óptimo.

$\eta(\boldsymbol{x}^*) = \sum_{j\in\mathcal{J}} w_j\frac{p_j}{s_{x_j^*}}$: suma ponderada de tiempos de proceso en la solución $\boldsymbol{x}^*$.

La solución jump-óptima es representada por el vector $\boldsymbol{x}$. Donde x_j indica la máquina donde es asignada la tarea j, es decir, $x_j = i$ si la tarea j es programada en la máquina i en el óptimo local. Adicionalmente se define:

$N_i(\boldsymbol{x}) = \{j \in \mathcal{J} : x_j = i\}$: el conjunto de tareas asignadas a la máquina i en $\boldsymbol{x}$;

$C_j(\boldsymbol{x})$: el tiempo de completación de la tarea j en la solución $\boldsymbol{x}$;

$C(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_jC_j(\boldsymbol{x})$: el costo de la solución jump-óptima $\boldsymbol{x}$.

$\eta(\boldsymbol{x}) = \sum_{j\in\mathcal{J}} w_j\frac{p_j}{s_{x_j}}$: suma ponderada de tiempos de proceso en la solución $\boldsymbol{x}$.

Con lo anterior y (6.1), es posible determinar los costos de las soluciones óptima y jump-óptima como:

$$C(\boldsymbol{x}^*) = \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\prec j}} w_j\frac{p_k}{s_{x_j^*}} = \eta(\boldsymbol{x}^*) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j^*}(\boldsymbol{x}^*)\\ k\succ j}} w_k\frac{p_j}{s_{x_j^*}} \tag{6.2}$$

$$C(\boldsymbol{x}) = \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\prec j}} w_j\frac{p_k}{s_{x_j}} = \eta(\boldsymbol{x}) + \sum_{j\in\mathcal{J}} \sum_{\substack{k\in N_{x_j}(\boldsymbol{x})\\ k\succ j}} w_k\frac{p_j}{s_{x_j}} \tag{6.3}$$

En la figura 4.1 se presenta un esquema de la movida de jump para la tarea j. Al asignar la tarea j a la máquina x_j^*, se produce un cambio en su tiempo de completación, este estará determinado por la velocidad de la máquina $s_{x_j^*}$, y la suma entre su tiempo de proceso (p_j) y el tiempo de proceso de todas las tareas asignadas a la máquina x_j^*, que preceden a la tarea j. Al retirar la tarea j de la máquina x_j, el tiempo de completación de todas las tareas que suceden a la tarea j se reduce en $\frac{p_j}{s_{x_j}}$. Al asignar la tarea j a la máquina x_j^*, el tiempo de completación de todas las tareas secuenciadas que suceden a la tarea j aumenta en $\frac{p_j}{s_{x_j^*}}$.

Para determinar el factor de aproximación de las soluciones jump-óptimas, se usará la cota superior establecida por el Lema 5.1. Para establecer una cota inferior del factor de aproximación, se presenta una instancia del problema, que expone un peor caso para los vecindarios de jump, exjump y swap locales. La construcción de esta instancia se basa en los resultados presentados por Caragiannis et al. [18], para juegos de balanceo descentralizado de carga de trabajo en el problema $R||\sum w_jC_j$.

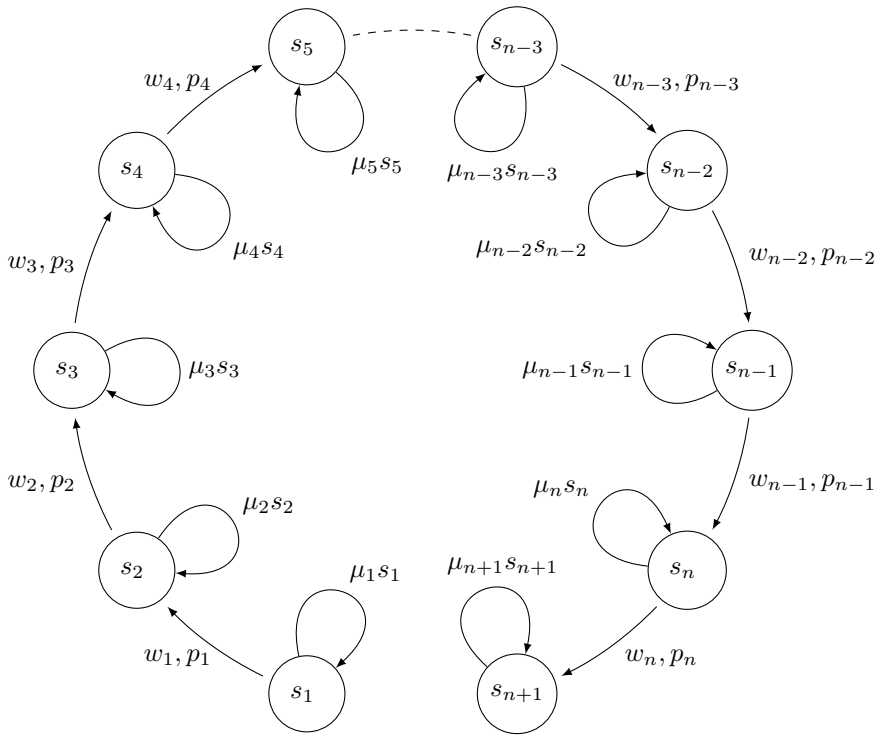
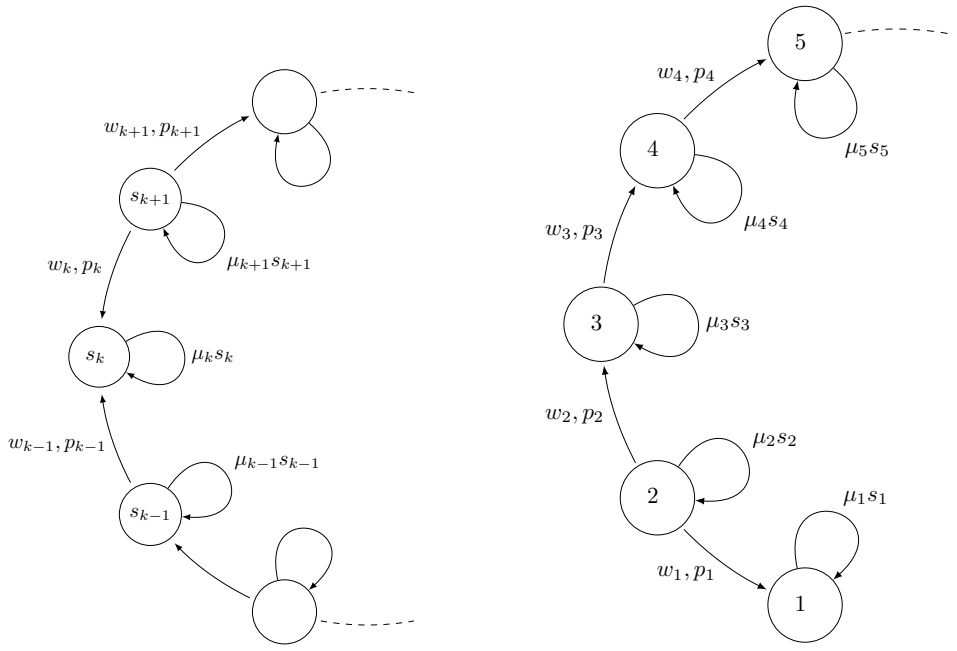


Figura 6.1: Instancia peor caso $RQ||\sum w_jC_j$

Lema 6.1 *El factor de aproximación de una solución jump-óptima, exjump-óptima o swap-óptima, para el problema $RQ||\sum w_jC_j$, es al menos* 2,618.

Demostración. Sea $\mathcal{I}$ una instancia con $m = n + 1$ máquinas y dos conjuntos de tareas, $\mathcal{J}_1$ y $\mathcal{J}_2$. La velocidad de la máquina i es s_i. Donde $s_1 = 1$, $s_i = \phi^{4-2i}$, para $i = 2, \ldots, n + 1$. El conjunto $\mathcal{J}_1$ tiene n tareas, el peso de la tarea j es $w_j = \phi^{2-j}$, $j = 1, \ldots, n$; con $\phi = \frac{1+\sqrt{5}}{2}$, el número áureo. La tarea j requiere un tiempo de proceso $p_j = \phi^{2-j}$, $j = 1, \ldots, n$, y puede



(a) $k \geq 2$ (b) $k = 1$

Figura 6.2: Movidas factibles para peor caso $RQ||\sum w_j C_j$

ser procesada por las máquinas j y $j+1$.
El conjunto $\mathcal{J}_2$ tiene $n+1$ tareas, el peso de cada tarea en este conjunto es unitario. La tarea j requiere un tiempo de proceso $\mu_j s_j,\ j = 1, \ldots, n+1$; con $\mu_1 > \mu_2 > \ldots > \mu_n > \mu_{n+1}$, tal que $\mu_j \to 0,\ \forall j = 1, \ldots, n+1$. La tarea j puede ser asignada únicamente a la máquina j.

En cualquier solución factible, la j-ésima tarea del conjunto $\mathcal{J}_2$ es asignada a la máquina j. En la solución óptima $\boldsymbol{x}^*$, la j-ésima tarea del conjunto $\mathcal{J}_1$ es asignada a la máquina j. Mientras que en la solución jump-óptima $\boldsymbol{x}$, es asignada a la máquina $j+1$.

En la figura 6.1 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen de cada arco, indica la máquina en la cual es asignada la tarea en la solución óptima $\boldsymbol{x}^*$. El destino de cada arco, indica la máquina en la cual es asignada la tarea en la solución local $\boldsymbol{x}$. Las tareas del conjunto $\mathcal{J}_2$ siempre serán secuenciadas primero, ya que su cociente de Smith (CS) es $\frac{1}{\mu_j s_j} \to \infty,\ \forall j \in \mathcal{J}_2$. Mientras que el CS de las tareas del conjunto $\mathcal{J}_1$ es 1.

En la figura 6.2 se presentan las movidas factibles para la solución jump-óptima. En la solución $\boldsymbol{x}$, sólo las tareas $j \in \mathcal{J}_1$ pueden ser movidas. Para verificar que $\boldsymbol{x}$ es un óptimo local de $\mathcal{I}$, se usarán las expresiones (5.4) y (5.5), de tal manera que se cumpla la condición de optimalidad local (5.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}_1$.

Para $k \geq 2$ (figura 6.2a), al mover la tarea k de la máquina $k+1$ a la máquina k, se genera un ahorro $\delta_k = \frac{w_k}{s_{k+1}}(\mu_{k+1}s_{k+1} + p_k) = w_k\left(\mu_{k+1} + \frac{p_k}{s_{k+1}}\right) = w_k\left(\mu_{k+1} + \phi^k\right)$. Al asignar la tarea k a la máquina k, se genera un incremento en $C(\boldsymbol{x})$, este es $\delta'_k = \frac{w_k}{s_k}(\mu_k s_k + p_{k-1} + p_k) =$

$w_k\left(\mu_k + \frac{\phi^{3-k}+\phi^{2-k}}{\phi^{4-2k}}\right) = w_k\left(\mu_k + \phi^k\right)$. Como $\mu_{k+1} < \mu_k$, se tiene que $\delta_k < \delta'_k$. Por lo tanto, mover la tarea k a la máquina k no reduce el valor de $C(\boldsymbol{x})$. Entonces, no se realizan movidas para la tarea k, $k \geq 2$.
Para $k = 1$ (figura 6.2b), al mover la tarea 1 de la máquina 2 a la máquina 1, se genera un ahorro $\delta_1 = \frac{w_1}{s_2}(\mu_2 s_2 + p_1) = w_1(\mu_2 + p_1)$. Al asignar la tarea 1 a la máquina 1, se genera un incremento en $C(\boldsymbol{x})$, este es $\delta'_1 = \frac{w_1}{s_1}(\mu_1 s_1 + p_1) = w_1(\mu_1 + p_1)$. Como $\mu_2 < \mu_1$, se tiene que $\delta_1 < \delta'_1$. Por lo tanto, mover la tarea 1 a la máquina 1 no reduce el valor de $C(\boldsymbol{x})$.
De manera similar, se puede verificar que el óptimo global es también un óptimo local.

Sea $\Xi = \sum_{j\in\mathcal{J}_2} \mu_j$; $\Pi_1 = \sum_{j=1}^{n} w_j\mu_j$; $\Pi_2 = \sum_{j=1}^{n} w_j\mu_{j+1}$.
El costo de las soluciones óptima[1] y jump-óptima es:

$$\begin{aligned}
C(\boldsymbol{x}^*) &= \sum_{j=1}^{n+1}\frac{\mu_j s_j}{s_j} + \sum_{j=1}^{n}\frac{w_j(\mu_j s_j + p_j)}{s_j} \\
&= \Xi + \Pi_1 + \sum_{j=1}^{n}\frac{w_j p_j}{s_j} \\
&= \Xi + \Pi_1 + \frac{w_1 p_1}{s_1} + \sum_{j=2}^{n}\frac{w_j p_j}{s_j} \\
&= \Xi + \Pi_1 + \phi^2 + \sum_{j=2}^{n}\frac{\phi^{2-j}\phi^{2-j}}{\phi^{4-2j}} \\
&= \Xi + \Pi_1 + \phi^2 + n - 1 \\
&= \Xi + \Pi_1 + \phi + n
\end{aligned}$$

$$\begin{aligned}
C(\boldsymbol{x}) &= \sum_{j=1}^{n+1}\frac{\mu_j s_j}{s_j} + \sum_{j=1}^{n}\frac{w_j(\mu_{j+1} s_{j+1} + p_j)}{s_{j+1}} \\
&= \Xi + \Pi_2 + \sum_{j=1}^{n}\frac{w_j p_j}{s_{j+1}} \\
&= \Xi + \Pi_2 + \sum_{j=1}^{n}\frac{\phi^{2-j}\phi^{2-j}}{\phi^{2-2j}} \\
&= \Xi + \Pi_2 + \sum_{j=1}^{n}\phi^2 \\
&= \Xi + \Pi_2 + n\phi^2
\end{aligned}$$

Luego, el factor de aproximación de un jump-óptimo es:

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{n\phi^2 + \Xi + \Pi_2}{n + \phi + \Xi + \Pi_1} = \frac{\phi^2 + \frac{\Xi+\Pi_2}{n}}{1 + \frac{\phi+\Xi+\Pi_1}{n}}$$

Para n suficientemente grande y cualquier $\varepsilon > 0$, se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \phi^2 - \varepsilon$.

[1] se usa $\phi = \phi^2 - 1$

Como todas las movidas posibles en la solución $\boldsymbol{x}$, generan un deterioro en el costo $C(\boldsymbol{x})$, la solución $\boldsymbol{x}$ es también un exjump-óptimo.

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}_1$ tal que $j \neq k$, se tiene que $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo. □

Con los Lemas 5.1, 6.1 y otras consideraciones, se puede determinar un factor de aproximación ajustado para las soluciones jump-óptima, exjump-óptima y swap-óptima; Teoremas 6.2, 6.3 y 6.4 respectivamente.

Teorema 6.2 *El factor de aproximación de una solución jump-óptima para el problema* $RQ||\sum w_j C_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación. Luego, $\varphi^J \leq \phi^2$. Por otra parte, el Lema 6.1 establece una cota inferior para el factor de aproximación. Luego, $\varphi^J \geq \phi^2$. Por lo tanto, $\varphi^J = \phi^2$. □

Teorema 6.3 *El factor de aproximación de una solución exjump-óptima para el problema* $R||\sum w_j C_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^L el factor de aproximación de un exjump-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, toda solución exjump-óptima es también un jump-óptimo. Luego, $\varphi^L \leq \phi^2$. Por otra parte, el Lema 6.1 establece una cota inferior para el factor de aproximación. Luego, $\varphi^L \geq \phi^2$. Por lo tanto, $\varphi^L = \phi^2$. □

Teorema 6.4 *El factor de aproximación de una solución swap-óptima para el problema* $R||\sum w_j C_j$ *es* $\phi^2 \approx 2{,}618$.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 5.1 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq \phi^2$. Por otra parte, el Lema 6.1 establece una cota inferior para el factor de aproximación. Luego, $\varphi^S \geq \phi^2$. Por lo tanto, $\varphi^S = \phi^2$. □

6.3. Tiempo de completación

En notación de scheduling este problema es representado por $RQ||\sum C_j$, donde el peso de todas las tareas es unitario. Por el Lema 1.3, se tiene que este problema es equivalente al caso donde el peso es constante para todas las tareas, $w_j = w$ para todo $j \in \mathcal{J}$.

Para determinar el factor de aproximación de las soluciones jump-óptimas, se usará la cota superior establecida por el Lema 5.6. Para establecer una cota inferior del factor de aproximación, se presenta una instancia del problema, que expone un peor caso para los vecindarios de jump, exjump y swap locales.

Lema 6.5 *El factor de aproximación de una solución jump-óptima, exjump-óptima o swap-óptima, para el problema* $RQ||\sum C_j$, *es al menos* 2.

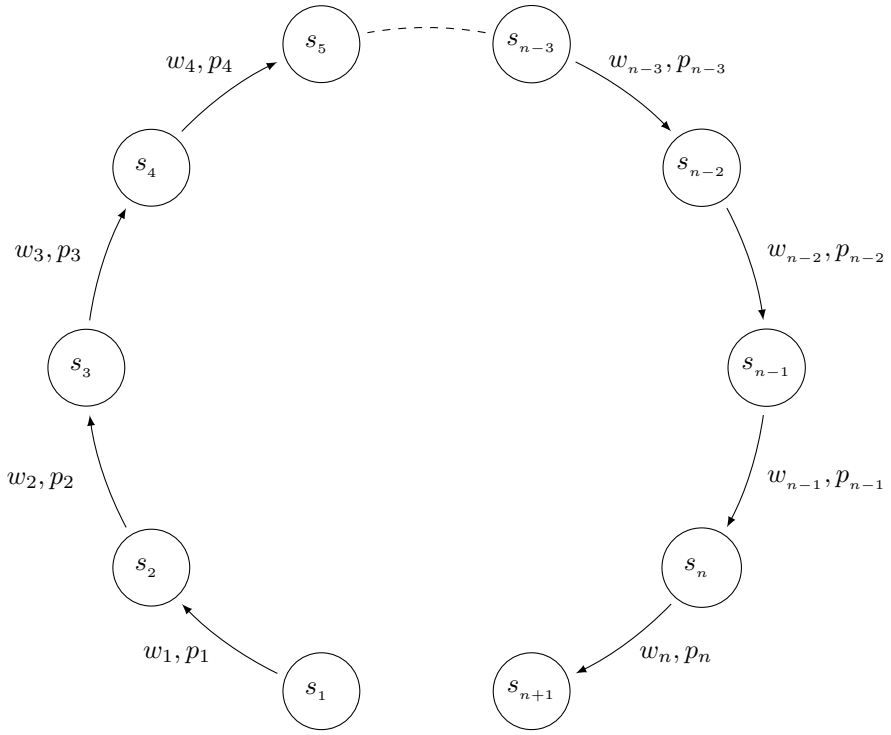


Figura 6.3: Instancia peor caso $RQ||\sum C_j$

Demostración. Sea $\mathcal{I}$ una instancia con n tareas y $m = n+1$ máquinas. El peso de las tareas y el tiempo de proceso requerido es $p_j = w_j = 1$, para $j = 1, \ldots, n$. La velocidad de las máquinas es $s_1 = 1 - \mu$; $s_i = (2(1-\mu))^{2-i}$, para $i = 2, \ldots, n+1$, con $\mu > 0$.
En la solución óptima $\boldsymbol{x}^*$, la tarea j es asignada a la máquina j. En cambio, en la solución jump-óptima $\boldsymbol{x}$, la tarea j es asignada a la máquina $j+1$.

En la figura 6.3 se presenta un esquema representativo de la instancia $\mathcal{I}$, donde cada máquina es representada por un nodo, y cada tarea es representada por un arco dirigido. El origen del arco k indica la máquina en la cual es asignada la tarea k en la solución óptima. En cambio, el destino del arco k indica la máquina en la cual es asignada la tarea k en la solución jump-óptima.

En la figura 6.4 se presentan las movidas factibles para la solución jump-óptima. Para la solución $\boldsymbol{x}$, la tarea k sólo puede ser movida a la máquina k. Para verificar que $\boldsymbol{x}$ es un óptimo local de $\mathcal{I}$ se usarán las expresiones (5.4) y (5.5), de tal manera que se cumpla la

condición de optimalidad local (5.6). Esto es, $\delta_j \leq \delta'_j$ para todo $j \in \mathcal{J}$. La secuencia de tareas para esta instancia se resuelve en forma arbitraria, ya que todas las tareas tienen tiempo de proceso unitario.

Para $k \geq 2$ (figura 6.4a), al mover la tarea k de la máquina $k+1$ a la máquina k, se genera un ahorro $\delta_k = \frac{1}{s_{k+1}} = (2(1-\mu))^{k-1}$. Al asignar la tarea k a la máquina k, se genera un incremento en $C(\boldsymbol{x})$, este es $\delta'_k = \frac{p_{k-1}+p_k}{s_k} = \frac{2}{(2(1-\mu))^{2-k}} = 2\,(2(1-\mu))^{k-2}$. Como $\mu > 0$, se tiene que $\delta_k < \delta'_k$. Por lo tanto, mover la tarea k a la máquina k no reduce el valor de $C(\boldsymbol{x})$. Entonces, no se realizan movidas para la tarea k, $k \geq 2$.
Para $k = 1$ (figura 6.4b), al mover la tarea 1 de la máquina 2 a la máquina 1, se genera un ahorro $\delta_1 = \frac{1}{s_2} = 1$. Al asignar la tarea 1 a la máquina 1, se genera un incremento en $C(\boldsymbol{x})$, este es $\delta'_1 = \frac{p_1}{s_1} = \frac{1}{1-\mu}$. Como $\mu > 0$, se tiene que $\delta_1 < \delta'_1$. Por lo tanto, mover la tarea 1 a la máquina 1 no reduce el valor de $C(\boldsymbol{x})$. Entonces, no se realizan movidas para la tarea 1.
De manera similar, se puede verificar que el óptimo global es también un óptimo local.

El costo de las soluciones óptima y jump-óptima es:

$$\begin{aligned}
C(\boldsymbol{x}^*) &= \sum_{j=1}^{n} \frac{1}{s_j} \\
&= \frac{1}{s_1} + \sum_{j=2}^{n} \frac{1}{s_j} \\
&= \frac{1}{1-\mu} + \sum_{j=2}^{n} \frac{1}{(2(1-\mu))^{2-j}} \\
&= \frac{1}{1-\mu} + \sum_{j=2}^{n} (2(1-\mu))^{j-2} \\
&= \frac{1}{1-\mu} + \sum_{j=0}^{n-2} (2(1-\mu))^{j} \\
&= \frac{(2(1-\mu))^{n-1}}{1-2\mu} - \frac{\mu}{(1-\mu)(1-2\mu)} \qquad (6.4)
\end{aligned}$$

$$\begin{aligned}
C(\boldsymbol{x}) &= \sum_{j=1}^{n} \frac{1}{s_{j+1}} \\
&= \sum_{j=1}^{n} \frac{1}{(2(1-\mu))^{1-j}} \\
&= \sum_{j=1}^{n} (2(1-\mu))^{j-1} \\
&= \sum_{j=0}^{n-1} (2(1-\mu))^{j} \\
&= \frac{(2(1-\mu))^{n}}{1-2\mu} - \frac{1}{1-2\mu} \qquad (6.5)
\end{aligned}$$

Multiplicando (6.4) y (6.5) por $R = \frac{1-2\mu}{(2(1-\mu))^{n-1}}$ se tiene:

$$\begin{aligned} C(\boldsymbol{x}^*)R &= 1 - \frac{\mu}{(1-\mu)(2(1-\mu))^{n-1}} \\ C(\boldsymbol{x}\ )R &= 2(1-\mu) - \frac{1}{(2(1-\mu))^{n-1}} \end{aligned}$$

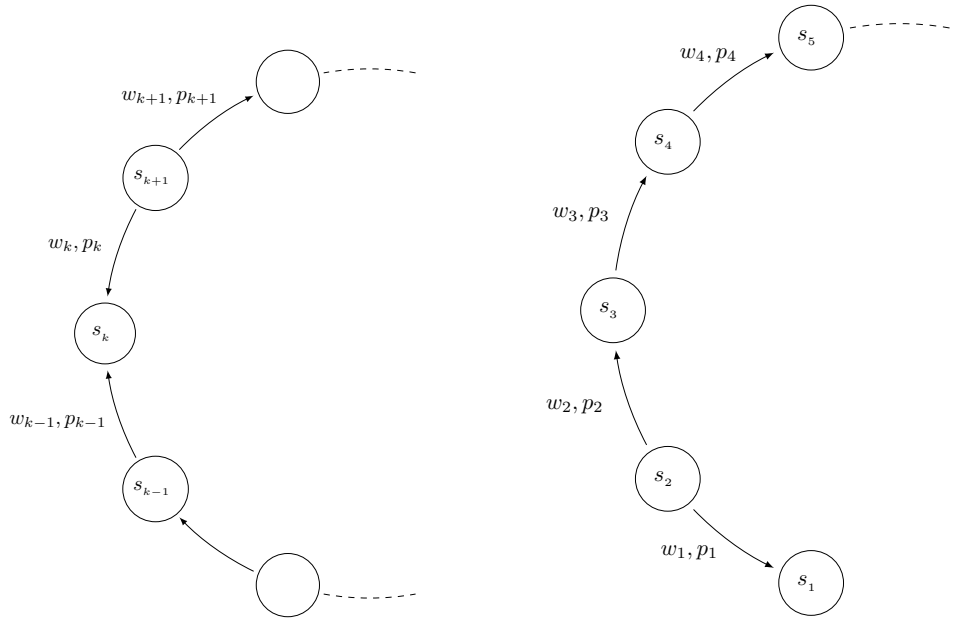
Luego, el factor de aproximación de un jump-óptimo es:

$$\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = \frac{2 - 2\mu - \frac{1}{(2(1-\mu))^{n-1}}}{1 - \frac{\mu}{(1-\mu)(2(1-\mu))^{n-1}}}$$

Para n suficientemente grande, $\mu \to 0$, y cualquier $\varepsilon > 0$, se tiene que $\frac{C(\boldsymbol{x})}{C(\boldsymbol{x}^*)} = 2 - \varepsilon$.

Como todas las movidas posibles en la solución $\boldsymbol{x}$, generan un deterioro en el costo $C(\boldsymbol{x})$, la solución $\boldsymbol{x}$ es también un exjump-óptimo.

En la instancia $\mathcal{I}$, para cada par de tareas $(j,k) \in \mathcal{J}_1$ tal que $j \neq k$, se tiene que $|\mathcal{M}_j \cap \mathcal{M}_k| \leq 1$. Por definición de vecindario swap, para que se pueda realizar una movida, se requiere que exista al menos una combinación de (j,k) tal que $|\mathcal{M}_j \cap \mathcal{M}_k| \geq 2$. Luego, no es posible realizar movidas de swap. Por lo tanto, la solución $\boldsymbol{x}$ es también un swap-óptimo. □



(a) $k \geq 2$ (b) $k = 1$

Figura 6.4: Movidas factibles para peor caso $RQ||\sum C_j$

Con los Lemas 5.6, 6.5, se puede determinar un factor de aproximación ajustado para las soluciones jump-óptima, exjump-óptima y swap-óptima; Teoremas 6.6, 6.7 y 6.8 respectivamente.

Teorema 6.6 *El factor de aproximación de una solución jump-óptima para el problema* $RQ||\sum C_j$ *es* 2.

Demostración. Sea φ^J el factor de aproximación de un jump-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación. Luego, $\varphi^J \leq 2$. Por otra parte, el Lema 6.5 establece una cota inferior para el factor de aproximación. Luego, $\varphi^J \geq 2$. Por lo tanto, $\varphi^J = 2$. □

Teorema 6.7 *El factor de aproximación de una solución exjump-óptima para el problema* $RQ||\sum C_j$ *es* 2.

Demostración. Sea φ^L el factor de aproximación de un exjump-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, toda solución exjump-óptima es también un jump-óptimo. Luego, $\varphi^L \leq 2$. Por otra parte, el Lema 6.5 establece una cota inferior para el factor de aproximación. Luego, $\varphi^L \geq 2$. Por lo tanto, $\varphi^L = 2$. □

Teorema 6.8 *El factor de aproximación de una solución swap-óptima para el problema* $RQ||\sum C_j$ *es* 2.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Lema 5.6 establece una cota superior para el factor de aproximación de un jump-óptimo. Pero, por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 2$. Por otra parte, el Lema 6.5 establece una cota inferior para el factor de aproximación. Luego, $\varphi^S \geq 2$. Por lo tanto, $\varphi^S = 2$. □

6.4. Tiempo ponderado de completación con tareas unitarias

En notación de scheduling este problema es representado por $RQ|p_j = 1|\sum w_j C_j$, donde el tiempo de proceso de todas las tareas es unitario.

Para determinar el factor de aproximación de las soluciones jump-óptimas, se demostrará que este problema es equivalente al problema $RQ||\sum C_j$ presentado en la Sección 6.3. Por lo tanto, los soluciones jump-óptimas del problema $RQ|p_j = 1|\sum w_j C_j$ tienen el mismo factor de aproximación del problema $RQ||\sum C_j$. Luego, estos resultados son extendidos a las soluciones exjump-óptimas y swap-óptimas.

Definición 6.9 *Sea* $\mathcal{I}$ *una instancia de* $RQ||\sum C_j$ *con tiempos de proceso* $q_1 \leq q_2 \leq \cdots \leq q_n$. *Se define la transformada* $\Upsilon : \mathcal{I} \to \mathcal{I}'$, *tal que* $\Upsilon(\mathcal{I})$ *genera una instancia* $\mathcal{I}'$ *del problema* $RQ|p_j = 1|\sum w_j C_j$ *con los mismos datos de* $\mathcal{I}$. *Esto es, para todo* $j \in \mathcal{J}$ *se hace* $w'_j = q_j$ *y* $p'_j = 1$. *Se puede notar que al aplicar la transformada inversa a* $\mathcal{I}'$, *se obtiene la instancia* $\mathcal{I}$.

Lema 6.10 *Sea $\mathcal{I}$ una instancia de $RQ||\sum C_j$, $\boldsymbol{z}$ una solución factible de $\mathcal{I}$, y sea $\mathcal{I}'$ la instancia obtenida al aplicar la transformada Υ sobre $\mathcal{I}$. La solución $\boldsymbol{z}$ tiene el mismo costo en ambas instancias, $\mathcal{I}'$ y $\mathcal{I}$.*

Demostración. Para simplificar las expresiones se considerará, sin pérdida de generalidad, que $q_1 < q_2 < \ldots < q_n$. De (6.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} w_j \frac{p_k}{s_i}$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{z})\right)$ para representar el costo de la solución $\boldsymbol{z}$ en la instancia $\mathcal{I}'$.
En la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{j \in \mathcal{J}} C_j(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \preceq j}} \frac{q_k}{s_i}$$

Para la instancia $\mathcal{I}$, el secuenciamiento de las tareas en $\boldsymbol{z}$ es determinado por la regla SPT. Luego, se puede sustituir el símbolo $\preceq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}(\boldsymbol{z})\right) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \leq j}} \frac{q_k}{s_i}$$

Por otra parte, de (6.1), se tiene que el costo de la solución $\boldsymbol{z}$ es:

$$C(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succeq j}} w_k \frac{p_j}{s_i}$$

En la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$ para todo $j \in \mathcal{J}$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{j \in \mathcal{J}} w_j C_j(\boldsymbol{z}) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \succeq j}} \frac{q_k}{s_i}$$

Para la instancia $\mathcal{I}'$, el secuenciamiento es determinado por la regla WSPT. Como en la instancia $\mathcal{I}'$ el tiempo de proceso de cada tarea es unitario, la regla WSPT es equivalente a secuenciar las tareas en orden decreciente de acuerdo al peso de las tareas ($w_j = q_j$). Luego, se puede sustituir el símbolo $\succeq$ por $\leq$. Entonces:

$$C\left(\mathcal{I}'(\boldsymbol{z})\right) = \sum_{i \in \mathcal{M}} \sum_{j \in N_i(\boldsymbol{z})} \sum_{\substack{k \in N_i(\boldsymbol{z}) \\ k \leq j}} \frac{q_k}{s_i}$$

Finalmente, se tiene que $C\left(\mathcal{I}(\boldsymbol{z})\right) = C\left(\mathcal{I}'(\boldsymbol{z})\right)$. □

Lema 6.11 *Sea $\boldsymbol{x}$ una solución jump-óptima de una instancia $\mathcal{I}$ del problema $RQ||\sum C_j$. Luego, $\boldsymbol{x}$ también es un jump-óptimo para la instancia $\mathcal{I}' = \Upsilon(\mathcal{I})$ del problema $RQ|p_j = 1|\sum w_j C_j$.*

Demostración. De la condición de optimalidad local (5.6) se tiene

$$w_j C_j(\boldsymbol{x}) + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ j}} w_k \frac{p_j}{s_{x_j}} \leq w_j \frac{p_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec j}} w_j \frac{p_k}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ j}} w_k \frac{p_j}{s_{x_j^*}} \ , \forall j \in \mathcal{J}$$

En lo que continúa, se usará $C\left(\mathcal{I}(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}$; y $C\left(\mathcal{I}'(\boldsymbol{x})\right)$ para representar el costo de la solución $\boldsymbol{x}$ en la instancia $\mathcal{I}'$. Para la instancia $\mathcal{I}$, se tiene que $p_j = q_j$ y $w_j = 1$, para todo $j \in \mathcal{J}$. Mientras que para la instancia $\mathcal{I}'$, se tiene que $p_j = 1$ y $w_j = q_j$, para todo $j \in \mathcal{J}$.
El objetivo es probar que $\boldsymbol{x}$ satisface la condición de optimalidad local para $\mathcal{I}'$. Es decir:

$$q_j C_j(\mathcal{I}'(\boldsymbol{x})) + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{q_k}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec j}} \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ j}} \frac{q_k}{s_{x_j^*}} \ , \forall j \in \mathcal{J}$$

Pero $C_j(\mathcal{I}'(\boldsymbol{x})) = \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{p_k}{s_{x_j}} = \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{1}{s_{x_j}}$, entonces

$$\sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{q_j}{s_{x_j}} + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{q_k}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec j}} \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ j}} \frac{q_k}{s_{x_j^*}} \ , \forall j \in \mathcal{J}$$

Para la instancia $\mathcal{I}'$, la secuencia de las tareas es definida por la regla WSPT, pero como las tareas tienen tiempo de proceso unitario, la secuencia de las tareas se define en orden decreciente según el peso de las tareas. Por lo tanto, se puede sustituir el símbolo $\succ$ por $<$ y el símbolo $\prec$ por $>$. Entonces la condición de optimalidad en $\mathcal{I}'$ es:

$$\sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \geq j}} \frac{q_j}{s_{x_j}} + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k < j}} \frac{q_k}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k > j}} \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k < j}} \frac{q_k}{s_{x_j^*}} \ , \forall j \in \mathcal{J} \tag{6.6}$$

Por otra parte, dado que $\boldsymbol{x}$ es un jump-local de $\mathcal{I}$, se sabe que:

$$C_j(\mathcal{I}(\boldsymbol{x})) + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{q_j}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec j}} \frac{q_k}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ j}} \frac{q_j}{s_{x_j^*}} \ , \forall j \in \mathcal{J}$$

Pero $C_j(\mathcal{I}(\boldsymbol{x})) = \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{p_k}{s_{x_j}} = \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{q_k}{s_{x_j}}$, entonces

$$\sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \preceq j}} \frac{q_k}{s_{x_j}} + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \succ j}} \frac{q_j}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \prec j}} \frac{q_k}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k \succ j}} \frac{q_j}{s_{x_j^*}} \ , \forall j \in \mathcal{J}$$

Para la instancia $\mathcal{I}$, la secuencia de las tareas es definida por la regla SPT. Por lo tanto, se puede sustituir el símbolo $\succ$ por $>$ y el símbolo $\prec$ por $<$. Entonces la condición de optimalidad en $\mathcal{I}$ es:

$$\sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k \leq j}} \frac{q_k}{s_{x_j}} + \sum_{\substack{k \in N_{x_j}(\boldsymbol{x}) \\ k > j}} \frac{q_j}{s_{x_j}} \leq \frac{q_j}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k < j}} \frac{q_k}{s_{x_j^*}} + \sum_{\substack{k \in N_{x_j^*}(\boldsymbol{x}) \\ k > j}} \frac{q_j}{s_{x_j^*}} \ , \forall j \in \mathcal{J} \tag{6.7}$$

Finalmente, por (6.6) y (6.7), se tiene que la condición de optimalidad es la misma para ambas instancias. Por lo tanto, si $\boldsymbol{x}$ es un jump-óptimo de la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. □

Teorema 6.12 *El factor de aproximación de una solución jump-óptima para el problema* $RQ|p_j = 1|\sum w_j C_j$ *es* 2.

Demostración. Sea $\mathcal{I}$ una instancia del problema $RQ||\sum C_j$, $\boldsymbol{x}$ una solución jump-óptima de $\mathcal{I}$, $\mathcal{I}'$ la instancia del problema $RQ|p_j = 1|\sum w_j C_j$ obtenida al aplicar la transformada Υ sobre $\mathcal{I}$.
Por el Lema 6.11, si $\boldsymbol{x}$ es un jump-óptimo para la instancia $\mathcal{I}$, también lo es para la instancia $\mathcal{I}'$. Por el Lema 6.10, se tiene que el costo de cualquier solución es el mismo en ambas instancias, $\mathcal{I}$ y $\mathcal{I}'$. Luego, se tiene que para cualquier instancia $\mathcal{I}'$ el factor de aproximación es el mismo que se tiene para la instancia $\mathcal{I}$. Entonces, se tiene que el problema $RQ|p_j = 1|\sum w_j C_j$ tiene el mismo factor de aproximación que el problema $RQ||\sum C_j$, el cual es establecido por el Teorema 6.6. □

Teorema 6.13 *El factor de aproximación de una solución exjump-óptima para el problema* $RQ|p_j = 1|\sum w_j C_j$ *es* 2.

Demostración. Sea φ^L el factor de aproximación de un exjump-óptimo. El Teorema 6.12 establece el factor de aproximación de un jump-óptimo. Por definición, toda solución exjump-óptima es también un jump-óptimo. Luego, $\varphi^L \leq 2$. Por otra parte, la instancia presentada en el Lema 6.5 es un caso particular del problema $RQ|p_j = 1|\sum w_j C_j$, y establece una cota inferior para el factor de aproximación. Luego, $\varphi^L \geq 2$. Por lo tanto, $\varphi^L = 2$. □

Teorema 6.14 *El factor de aproximación de una solución swap-óptima para el problema* $RQ|p_j = 1|\sum w_j C_j$ *es* 2.

Demostración. Sea φ^S el factor de aproximación de un swap-óptimo. El Teorema 6.12 establece el factor de aproximación de un jump-óptimo. Por definición, un swap-óptimo es también un jump-óptimo. Luego, $\varphi^S \leq 2$. Por otra parte, la instancia presentada en el Lema 6.5 es un caso particular del problema $RQ|p_j = 1|\sum w_j C_j$, y establece una cota inferior para el factor de aproximación. Luego, $\varphi^S \geq 2$. Por lo tanto, $\varphi^S = 2$. □

6.5. Tiempo de completación con tareas unitarias

En notación de scheduling este problema es representado por $RQ|p_j = 1|\sum C_j$, donde el peso y el tiempo de proceso de todas las tareas es unitario. Este problema es un caso particular del problema $RQ||\sum C_j$, y éste a su vez es un caso particular de $R||\sum C_j$. Por lo tanto, el Teorema 5.6 establece una cota superior para el factor de aproximación de un jump-óptimo. Por otra parte, la instancia presentada en el Lema 6.5 tiene vectores $\boldsymbol{w}$ y $\boldsymbol{p}$ unitarios, con lo que se establece una cota inferior para este problema.

Teorema 6.15 *El factor de aproximación de una solución jump-óptima para el problema* $RQ|p_j = 1|\sum C_j$ *es* 2.

Demostración. Ver demostración del Teorema 6.6. □

Teorema 6.16 *El factor de aproximación de una solución exjump-óptima para el problema* $RQ|p_j = 1|\sum C_j$ *es* 2.

Demostración. Ver demostración del Teorema 6.7. □

Teorema 6.17 *El factor de aproximación de una solución swap-óptima para el problema* $RQ|p_j = 1|\sum C_j$ *es* 2.

Demostración. Ver demostración del Teorema 6.8. □

Capítulo 7

Conclusiones y problemas abiertos

En este trabajo se estudia la calidad de los óptimos locales para el problema de minimizar el tiempo de completación ponderado en ambientes de máquinas en paralelo. Se determinaron factores de aproximación para los óptimos locales de varios vecindarios.

Para el problema de máquinas paralelas no-relacionadas ($R||\sum w_jC_j$) y máquinas paralelas uniformes ($RQ||\sum w_jC_j$), se determinó un factor de aproximación ajustado de 2,618. Lo que indica que los óptimos locales para vecindarios de jump y swap se encuentran alejados a lo más en 2,618 con respecto al óptimo. Para los problemas no ponderados $R||\sum C_j$ y $RQ||\sum w_jC_j$, se determinó un factor de aproximación ajustado de 2. Lo que indica que los óptimos locales para vecindarios de jump y swap se encuentran alejados a lo más en 2 con respecto al óptimo. Para los casos particulares $RQ|p_j = 1|\sum w_jC_j$ y $RQ|p_j = 1|\sum C_j$ se determinó el mismo resultado. También se determinó que considerar una extensión del vecindario jump, donde el conjunto de soluciones candidatas incluye aquellas movidas que tienen el mismo costo de la solución incumbente, no mejora el factor de aproximación.

Para el problema de máquinas paralelas restringidas idénticas ($RP||\sum w_jC_j$), se determinó que el factor de aproximación de un óptimo local para los vecindarios jump o swap se encuentra entre 1,75 y 1,809, mientras que para el caso no ponderado ($RP||\sum C_j$) se encentra entre 1,533 y 1,618. Estas brechas constituyen los primeros dos problemas abiertos. También se determinó que los óptimos locales para los vecindarios de jump y swap del problema $RP|p_j = 1|\sum w_jC_j$ tiene el mismo factor de aproximación que el problema $RP||\sum C_j$. Para el problema $RP|p_j = 1|\sum C_j$ se presenta una conjetura sobre el factor de aproximación. Esta conjetura constituye el tercer problema abierto.

Para el problema de máquinas idénticas, no se logró establecer nuevos resultados con respecto al factor de aproximación constante. Sin embargo se logró establecer factores de aproximación no constantes para instancias de los problemas $P||\sum w_jC_j$ y $P||\sum C_j$. Por lo tanto, acotar la brecha entre las cotas superior e inferior del factor de aproximación constituye el cuarto problema abierto.

Bibliografía

[1] Emile Aarts and Jan K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[2] Fidaa Abed. *Coordinating selfish players in scheduling games*. PhD thesis, Saarbrücken, Universität des Saarlandes, Diss., 2015.

[3] Fidaa Abed, José R. Correa, and Chien-Chung Huang. Optimal coordination mechanisms for multi-job scheduling games. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014: 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 13–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[4] Ravindra K Ahuja, Özlem Ergun, James B Orlin, and Abraham P Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1):75–102, 2002.

[5] Eric Angel. A survey of approximation results for local search algorithms. In Evripidis Bampis, Klaus Jansen, and Claire Kenyon, editors, *Efficient Approximation and Online Algorithms*, volume 3484 of Lecture Notes in Computer Science, pages 30–73. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.

[6] Baruch Awerbuch, Yossi Azar, Yossi Richter, and Dekel Tsur. Tradeoffs in worst-case equilibria. *Theoretical Computer Science*, 361(2–3):200–209, 2006.

[7] Meral Azizoglu and Omer Kirca. On the minimization of total weighted flow time with identical and uniform parallel machines. *European Journal of Operational Research*, 113(1):91–100, 1999.

[8] Kenneth R. Baker. *Introduction to sequencing and scheduling*. John Wiley & Sons, 1974.

[9] Kenneth R. Baker and Dan Trietsch. *Principles of sequencing and scheduling*. John Wiley & Sons, 2013.

[10] Nikhil Bansal, Aravind Srinivasan, and Ola Svensson. Lift-and-round to improve weighted completion time on unrelated machines. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 156–167, New York, NY, USA, 2016. ACM.

[11] Peter Brucker. *Scheduling Algorithms*. Springer Science & Business Media, 2007.

[12] Peter Brucker, Johann Hurink, and Frank Werner. Improving local search heuristics for some scheduling problems. Part I. *Discrete Applied Mathematics*, 65(1-3):97–122, 1996.

[13] Peter Brucker, Johann Hurink, and Frank Werner. Improving local search heuristics for some scheduling problems. Part II. *Discrete Applied Mathematics*, 72(1-2):47–69, 1997.

[14] T. Brueggemann, J.L. Hurink, T. Vredeveld, and G.J. Woeginger. Very large-scale neighborhoods with performance guarantees for minimizing makespan on parallel machines. In C. Kaklamanis and M. Skutella, editors, *5th International Workshop on Approximation and Online Algorithms*, volume 4927 of *Lecture Notes in Computer Science*, pages 41–54, Berlin, February 2008. Springer Verlag.

[15] Tobias Brueggemann, Johann L. Hurink, and Walter Kern. Quality of move-optimal schedules for minimizing total weighted completion time. *Operations Research Letters*, 34(5):583–590, 2006.

[16] Tobias Brueggemann, Johann L. Hurink, Tjark Vredeveld, and Gerhard J. Woeginger. Exponential size neighborhoods for makespan minimization scheduling. *Naval Research Logistics (NRL)*, 58(8):795–803, 2011.

[17] James Bruno, Edward G. Coffman Jr., and Ravi Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387, 1974.

[18] Ioannis Caragiannis, Michele Flammini, Christos Kaklamanis, Panagiotis Kanellopoulos, and Luca Moscardelli. Tight bounds for selfish and greedy load balancing. *Algorithmica*, 61(3):606–637, 2011.

[19] Bo Chen. Tighter bound for multifit scheduling on uniform processors. *Discrete Applied Mathematics*, 31(3):227 – 260, 1991.

[20] Yookun Cho and Sartaj Sahni. Bounds for list schedules on uniform processors. *SIAM Journal on Computing*, 9(1):91–103, 1980.

[21] Richard Cole, José R. Correa, Vasilis Gkatzelis, Vahab Mirrokni, and Neil Olver. Decentralized utilitarian mechanisms for scheduling games. *Games and Economic Behavior*, 92:306 – 326, 2015.

[22] RW. Conway, W.L Maxwell, and LW. Miller. *Theory of scheduling*. Addison-Wesley, Reading, MA, 1967.

[23] José R. Correa and Maurice Queyranne. Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. *Naval Research Logistics (NRL)*, 59(5):384–395, 2012.

[24] Artur Czumaj and Berthold Vöcking. Tight bounds for worst-case equilibria. *ACM Trans. Algorithms*, 3(1):4:1–4:17, February 2007.

[25] Ernest Davis and Jeffrey M. Jaffe. Algorithms for scheduling tasks on unrelated processors. *Journal of the ACM (JACM)*, 28(4):721–736, 1981.

[26] Gregory Dobson. Scheduling independent tasks on uniform processors. *SIAM Journal on Computing*, 13(4):705–716, 1984.

[27] Willard L. Eastman, Shimon Even, and IM. Isaacs. Bounds for the optimal scheduling of n jobs on m processors. *Management science*, 11(2):268–279, 1964.

[28] Salah E. Elmaghraby and Sung H. Park. Scheduling jobs on a number of identical machines. *AIIE transactions*, 6(1):1–13, 1974.

[29] Leah Epstein and Jiri Sgall. Approximation schemes for scheduling on uniformly related and identical parallel machines. *Algorithmica*, 39(1):43–57, 2004.

[30] Greg Finn and Ellis Horowitz. A linear time approximation algorithm for multiprocessor scheduling. *BIT Numerical Mathematics*, 19(3):312–320, 1979.

[31] Antonio Frangioni, Emiliano Necciari, and Maria Grazia Scutella. A multi-exchange neighborhood for minimum makespan parallel machine scheduling problems. *Journal of Combinatorial Optimization*, 8(2):195–220, 2004.

[32] Donald K. Friesen. Tighter bounds for LPT scheduling on uniform processors. *SIAM Journal on Computing*, 16(3):554–560, 1987.

[33] Donald K. Friesen and Michael A. Langston. Bounds for multifit scheduling on uniform processors. *SIAM Journal on Computing*, 12(1):60–70, 1983.

[34] Martin Gairing, Thomas Lücking, Marios Mavronicolas, and Burkhard Monien. The price of anarchy for restricted parallel links. *Parallel Processing Letters*, 16(01):117–131, 2006.

[35] Michael R. Garey and David S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, 4(4):397–411, 1975.

[36] Michael R. Garey and David S. Johnson. "Strong" NP-Completeness results: Motivation, examples, and implications. *Journal of the ACM (JACM)*, 25(3):499–508, 1978.

[37] Michael R. Garey and David S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completness*. WH Freeman and Co, San Francisco, 1979.

[38] Teofilo Gonzalez, Oscar H. Ibarra, and Sartaj Sahni. Bounds for LPT schedules on uniform processors. *SIAM Journal on Computing*, 6(1):155–166, 1977.

[39] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581, 1966.

[40] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

[41] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and AHG. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.

[42] Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

[43] Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM journal on computing*, 17(3):539–551, 1988.

[44] R.P. Hoeksma and M.J. Uetz. The price of anarchy for minsum related machine scheduling. In R. Solis-Oba and G. Persiano, editors, *9th International Workshop on Approximation and Online Algorithms, WAOA 2011, Saarbrücken, Germany*, volume 7164 of *Lecture Notes in Computer Science*, pages 261–273, Heidelberg, 2012. Springer Verlag.

[45] WA. Horn. Minimizing average flow time with parallel machines. *Operations research*, 21(3):846–847, 1973.

[46] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM (JACM)*, 23(2):317–327, 1976.

[47] Ellis Horowitz and Sartaj Sahni. *Fundamentals of computer algorithms*. Computer Science Press, 1978.

[48] Cor AJ. Hurkens and Tjark Vredeveld. Local search for multiprocessor scheduling: how many moves does it take to a local optimum? *Operations Research Letters*, 31(2):137–141, 2003.

[49] Oscar H. Ibarra and Chul E. Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *Journal of the ACM (JACM)*, 24(2):280–289, 1977.

[50] Klaus Jansen and Lorant Porkolab. Improved approximation schemes for scheduling unrelated parallel machines. *Mathematics of Operations Research*, 26(2):324–338, 2001.

[51] Tsuyoshi Kawaguchi and Seiki Kyan. Worst case bound of an LRF schedule for the mean weighted flow-time problem. *SIAM Journal on Computing*, 15(4):1119–1129, 1986.

[52] Eugene L. Lawler, Jan Karel Lenstra, Alexander HG. Rinnooy Kan, and David B. Shmoys. Sequencing and scheduling: Algorithms and complexity. In SC. Graves, AHG. Rinnooy Kan, and PH. Zipkin, editors, *Handbooks in operations research and management science*, volume 4, chapter 9, pages 445–522. Elsevier, 1993.

[53] Jan Karel Lenstra, AHG. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of discrete mathematics*, 1:343–362, 1977.

[54] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3):259–271,

1990.

[55] Joseph Y-T. Leung and Chung-Lun Li. Scheduling with processing set restrictions: A survey. *International Journal of Production Economics*, 116(2):251–262, 2008.

[56] Wil Michiels, Emile Aarts, and Jan Korst. *Theoretical aspects of local search*. Springer Science & Business Media, 2007.

[57] Nanda Piersma and Wim van Dijk. A local search heuristic for unrelated parallel machine scheduling with efficient neighborhood search. *Mathematical and Computer Modelling*, 24(9):11–19, 1996.

[58] Michael L. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media, 2012.

[59] Diego Recalde, Cyriel Rutten, Petra Schuurman, and Tjark Vredeveld. Local search performance guarantees for restricted related parallel machine scheduling. *LATIN 2010: Theoretical Informatics*, pages 108–119, 2010.

[60] Cyriel Rutten, Diego Recalde, Petra Schuurman, and Tjark Vredeveld. Performance guarantees of jump neighborhoods on restricted related parallel machines. *Operations Research Letters*, 40(4):287–291, 2012.

[61] Sartaj K. Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM (JACM)*, 23(1):116–127, 1976.

[62] Andreas S. Schulz and Martin Skutella. Scheduling unrelated machines by randomized rounding. *SIAM Journal on Discrete Mathematics*, 15(4):450–469, 2002.

[63] Petra Schuurman and Tjark Vredeveld. Performance guarantees of local search for multiprocessor scheduling. *INFORMS Journal on Computing*, 19(1):52–63, 2007.

[64] Uwe Schwiegelshohn. An alternative proof of the Kawaguchi-Kyan bound for the largest-ratio-first rule. *Operations Research Letters*, 39(4):255–259, 2011.

[65] Martin Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM (JACM)*, 48(2):206–242, 2001.

[66] Martin Skutella and Gerhard J. Woeginger. A ptas for minimizing the total weighted completion time on identical parallel machines. *Mathematics of Operations Research*, 25(1):63–75, 2000.

[67] Wayne E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.

[68] Berthold Vöcking. Selfish load balancing. In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V. Vazirani, editors, *Algorithmic game theory*, chapter 20. Cambridge University Press Cambridge, New York, 2007.

[69] Scott Webster. New bounds for the identical parallel processor weighted flow time problem. *Management science*, 38(1):124–136, 1992.

[70] Scott Webster. Weighted flow time bounds for scheduling identical processors. *European Journal of Operational Research*, 80(1):103–111, 1995.