**UNIVERSIDAD CENTROCCIDENTAL**
**"LISANDRO ALVARADO"**

**Decanato de Ciencias y Tecnología**
**Licenciatura en Ciencias Matemáticas**


# " Algoritmos de aproximación para problemas de programación de tareas con un objetivo de tardanza ponderada total modificado "

Trabajo Especial de Grado presentado por

Br. Jaicer J. López R.

como requisito final
para obtener el título de Licenciado
en Ciencias Matemáticas
Área de Conocimiento: Optimización.
Tutor: MSc. Alí Duín

Barquisimeto, Venezuela. Noviembre de 2016

Universidad Centroccidental
"Lisandro Alvarado"
Decanato de Ciencias y Tecnología
Licenciatura en Ciencias Matemáticas

ACTA
TRABAJO ESPECIAL DE GRADO

Los suscritos miembros del Jurado designado por el Jefe del Departamento de Matemáticas del Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", para examinar y dictar el veredicto sobre el Trabajo Especial de Grado titulado:

" ALGORITMOS DE APROXIMACIÓN PARA PROBLEMAS DE PROGRAMACIÓN DE TAREAS CON UN OBJETIVO DE TARDANZA PONDERADA TOTAL MODIFICADO "

presentado por el ciudadano BR. JAICER J. LÓPEZ R. titular de la Cédula de Identidad No. 19.827.103, con el propósito de cumplir con el requisito académico final para el otorgamiento del título de Licenciado en Ciencias Matemáticas.

Luego de realizada la Defensa y en los términos que imponen los Lineamientos para el Trabajo Especial de Grado de la Licenciatura en Ciencias Matemáticas, se procedió a discutirlo con el interesado habiéndose emitido el veredicto que a continuación se expresa:

[1] _____

Con una calificación de _____ puntos.

En fe de lo expuesto firmamos la presente Acta en la Ciudad de Barquisimeto a los _____ días del mes de _____ de _____.

| | |
|---|---|
| _____ | _____ |
| TUTOR | FIRMA |
| _____ | _____ |
| PRINCIPAL | FIRMA |
| _____ | _____ |
| PRINCIPAL | FIRMA |

OBSERVACIONES:

_____

_____

_____

[1] Aprobado ó Reprobado

*Dedicado a mi familia. En especial a mis padres, esposa e hijos.*

# AGRADECIMIENTOS

Quisiera agradecer a todas las personas que colaboraron en mi formación tanto académica como personal. Primeramente agradecerle a Dios y a mi familia, a mis padres Josefina Rivero y Omar López, sin ellos no fuera sido posible este logro, a mis hermanos Jaiker y Jailer, a mi esposa Luz Alvarado, a mis dos hijos Jaicer Fabian y Adrian Jonás, los cuales siempre me dan ese impulso de seguir adelante y cumplir mis metas.

A mi tía Dilcia Leon que siempre estuvo en los momentos difíciles, a mis tías: Luisa, Elvira, Deisy y Elina. A mi cuñada Neidis Alvarado, a mi suegra Maria Rodriguez la cual me presto su apoyo con el computador para realizar este trabajo.

A los profesores de los cuales recibí una excelente formación, Eibar Hernández, Victor Caruci, Mario Rodriguez, Omar Rodriguez, Javier Hernández, Ebner Pineda, Julio Ysaccura, Hugo Lara, Maria Linares y a mi tutor Ali Duin, quien me presto su apoyo a pesar del momento difícil por el cual paso, a todos ellos muchas gracias.

También quiero agradecerles a los compañeros que conocí en el decanato y que también colaboraron en este logro, entre ellos, Oscar Sapienza, Eduardo Fuentes, Harry Oviedo, Cleiver Cordoba, Ronny Quintero, Diana Piñango, Gustavo Castillo, Yuribel Aguero, Hernan Teran, Naudy Gimenez, Samir Chirinos y a Aldemar Gómez por la colaboración prestada con el formato de tesis.

# Resumen

En este trabajo se aborda el estudio de los problemas de programación de tareas con el objetivo de la tardanza ponderada total. Dicho estudio está basado en el artículo *"Approximation algorithms for scheduling problems with a modified total weighted tardiness objective"* [1], de los autores Stavros G. Kolliopoulos y George Steiner. Se obtienen dos resultados importantes a partir de la modificación de la función objetivo y del uso de resultados previos para el problema con el objetivo de los tiempos de completación ponderados totales. Se describen de manera detallada los resultados obtenidos y se estudia un problema en particular.

# Índice general

# Índice de figuras

# Índice de cuadros

# INTRODUCCIÓN

Los modelos de programación de tareas constituyen una valiosa herramienta para la toma de decisiones en el sector productivo de la sociedad actual, sobre todo en aquellas empresas que producen bajo pedido. En dichas empresas se transforman materias primas en productos elaborados y es frecuente encontrar esquemas de producción bajo pedido y también esquemas mixtos, esto es, bajo pedido y por programas. Cuando se trabaja bajo pedido la empresa conoce de antemano la demanda futura y en consecuencia puede planificar exactamente como debe actuar, sin embargo la tarea no es trivial, más aún, usualmente es $\mathcal{NP}$-completo; Es justo aquí donde se inserta el trabajo de Kolliopolus y Steiner, específicamente en aquellos casos en donde la gerencia considere que es muy importante entregar los trabajo a tiempo o con el menor retraso posible.

Se presenta en este T.E.G. la interpretación de uno de los métodos de programación de tareas, precisamente en la situación planteada en el párrafo previo. La exposición se seccionó en dos capítulos, por motivos didácticos, aspecto que fue considerado durante el presente desarrollo. En el capítulo 1 se exponen los preliminares, una serie de conceptos y notaciones muy útiles en la comprensión tanto del problema como de los métodos, y también de la magnitud de la tarea, la cual parece fácil en las primeras de cambio, pero que al final resulta dura. Por este motivo se exponen, brevemente, algunos conceptos de complejidad algorítmica. En el segundo capítulo se describe el método en sí, el cual es en verdad una familia de métodos, basados en dos ideas principales:

1. Transformar el objetivo de tardanzas ponderadas por el de tiempos de completación ponderados.

2. Utilizar una relajación del problema mediante una formulación de programación lineal y partir de la solución de esta relajación conseguir una solución factible de muy buena calidad, aunque no necesariamente óptima.

Debe observarse que en la exposición original, Kolliopolus y Steiner, desarrollan el caso $1 \mid d_j = D \mid \sum_j (T_j + d_j)$, mientras que en este trabajo se desarrolló el caso $1 \mid prec \mid \sum_j (T_j + d_j)$, lo cual constituye un aporte al tema.

# Capítulo 1

# Preliminares

En este capítulo se describe de manera general, el problema de programación de tareas (en inglés **Scheduling problems**). Se dará una clasificación básica de los problemas de programación de tareas, la cual es utilizada ampliamente en la literatura. Se muestran un par de ejemplos del problema para mayor comprensión. Luego, se repasan algunos conceptos claves en lo que respecta a la complejidad computacional y se define tanto algoritmo de aproximación como relajación lineal. Por último se describe el algoritmo del elipsoide. La mayoría de las definiciones fueron extraídas de [2].

## 1.1. Problemas de programación de tareas

Cualquier problema de programación de tareas viene asociado con un conjunto finito de tareas y un conjunto finito de máquinas. Se denota al conjunto de todas las tareas como $N = \{1, \ldots, n\}$. Cada tarea $j$, $1 \leq j \leq n$, debe ser procesada en una de las $m\,(m \geq 1)$ máquinas.

**Definición 1.1.** Un **horario** es para cada tarea una asignación de uno o más intervalos de tiempo a una o más máquinas.

**Definición 1.2.** Un **horario** es **factible** si no hay dos intervalos de tiempo superpuestos en la misma máquina y si además, cumple con una serie de características para problemas específicos.

**Definición 1.3.** Un **horario** es **óptimo** si siendo factible, minimiza el criterio de optimalidad dado.

Los horarios pueden ser representados por los diagramas de Gantt como se muestra en la figura:

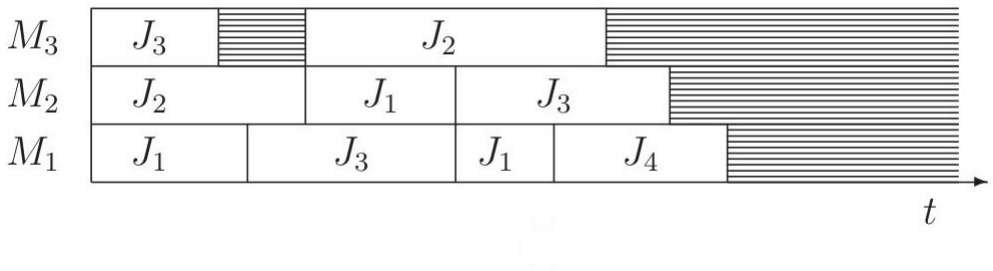


Figura 1.1: Diagrama de Gantt

**Definición 1.4.** Para un horario determinado de las tareas, la **Tardanza** de la tarea $j$ se define como $T_j := \text{máx}\{C_j - d_j, 0\}$, donde $C_j$ es el tiempo de completación de la tarea, es decir el instante donde la tarea $j$ termina de procesarse.

**Definición 1.5.** La **Anticipación** de la tarea $j$ en un horario $\sigma$ es definida como $E_j(\sigma) := \text{máx}\{d_j - C_j, 0\}$.

Toda tarea tiene los siguientes datos asociados a ella:

- **Tiempo de procesamiento** $(p_j)$ : Tiempo empleado por la máquina para procesar la tarea $j$.
- **Fecha de entrega** $(d_j)$ : Fecha pautada en la cual la tarea $j$ debería ser completada. Muchas veces, se permite la completación después de la fecha de entrega. Sin embargo una penalización debiera generarse por hacerlo.
- **Ponderación** $(w_j)$ : La ponderación de una tarea denota el valor relativo de una tarea con respecto a las otras.

Las clases de problemas de programación de tareas se especifican en términos de una clasificación de tres campos $\alpha \,|\, \beta \,|\, \gamma$, donde " $\alpha$ " especifica el entorno de la máquina, " $\beta$ " especifica las características de las tareas, y " $\gamma$ " denota el criterio de optimalidad. Dicho sistema de clasificación fue introducido por Graham y otros [3].

### 1.1.1. Entorno de la máquina $\alpha$

Los posibles entornos de máquinas que se presentan en este trabajo son los siguientes:

- **Una sola máquina** ($\alpha = 1$) : Aunque pueda parecer un caso muy especial, el estudio de estos problemas dará lugar a muchas técnicas útiles.

- **Máquinas paralelas idénticas** ($\alpha = P$) : En este caso tenemos $m$ máquinas idénticas y cualquier tipo de tarea se puede ejecutar en cualquier máquina con el mismo tiempo de procesamiento en cada una.

- **Máquinas uniformemente relacionadas** ($\alpha = Q$) : En este caso disponemos de $m$ máquinas y cualquier tipo de tarea se puede ejecutar en cualquier máquina. Sin embargo, cada máquina $i$ tiene una velocidad $s_i \geq 1$. El procesamiento de la tarea $j$ en la máquina $i$ requiere $p_j/s_j$ unidades de tiempo.

- **Máquinas no relacionadas** ($\alpha = R$) : En este caso tenemos $m$ máquinas y cualquier tarea se puede ejecutar en cualquier máquina. Sin embargo, cada tarea $j$ toma $p_{ij}$ unidades de tiempo de procesamiento si se asigna a la máquina $i$. $R_m$ significa máquinas no relacionadas cuyo número es fijo.

### 1.1.2. Características de las tareas $\beta$

- **Tiempo de liberación** ($\beta = r_j$) : Hay problemas en los que una tarea solo está disponible para su procesamiento después de un tiempo $r_j > 0$. A menos que se especifique, se supone que todas las tareas están disponibles desde el principio.

- **Restricciones de precedencia** ($\beta = prec$) : Muchas veces una tarea $j$ no puede ser procesada hasta que la tarea $k$ es completada. Tales restricciones se denominan restricciones de precedencia. Si la tarea $k$ precede a la tarea $j$ escribiremos $k \prec j$.

- **Interrupciones** ($\beta = pmtn$) : Una tarea se puede adelantar si no es necesario que se complete su procesamiento una vez que ha comenzado. Es decir puede ser interrumpida y reanudada más tarde. Por defecto, se supone que las interrupciones no están permitidas.

En general, todos los datos $p_j$, $r_j$, $d_j$, $w_j$ se suponen enteros.

### 1.1.3. Criterio de optimalidad $\gamma$

En la teoría de los problemas de programación de tareas se encuentran una gran cantidad de funciones objetivo, sin embargo en el presente trabajo la función objetivo séra la tardanza ponderada total, la cual se denota como sigue:

$$\sum_{j=1}^{n} w_j T_j.$$

Otras funciones objetivo que también aparecen, serán el tiempo de completación ponderado total:

$$\sum_{j=1}^{n} w_j C_j,$$

y la demora máxima

$$L_{\text{máx}} = \text{máx}\{L_j, j \in N\},$$

donde la demora de la tarea $j$ se define como $L_j := C_j - d_j$ .
El problema de programación de tareas correspondiente es encontrar un horario factible que satisfaga ciertas restricciones y que minimice el criterio de optimalidad dado.

### 1.1.4. Ejemplos

**Ejemplo 1.1.** Considere un aeropuerto concurrido, donde decenas de aviones aterrizan y despegan a cada minuto. Por el momento solo considere los aviones que aterrizan. Supongamos que el aeropuerto cuenta con $m$ compuertas, también supongamos que el avión $j$ aterriza en un tiempo $r_j$ y por otra parte le toma $p_j$ unidades de tiempo viajar desde la terminal hasta la compuerta $m$. Obviamente en cualquier momento pudieran coincidir dos aviones en una misma compuerta. ¿Como decidir a que avión enviar y a que compuerta de modo que el tiempo promedio tomado por los pasajeros para salir del avión es minimizado? .

**Ejemplo 1.2.** Para ilustrar la notación de tres campos $\alpha \,|\, \beta \,|\, \gamma$ se muestra el siguiente ejemplo. $1 \,|\, r_i; pmtn \,|\, L_{\text{máx}}$ el problema de encontrar un horario en el cual se permiten las interrupciones en una máquina para una serie de tareas con tiempos de liberación dados $r_i \neq 0$, tal que la demora máxima es minimizada. Los datos de entrada del problema así como un horario factible se presentan a continuación:

| $i$ | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| $p_i$ | 2 | 1 | 2 | 2 |
| $r_i$ | 1 | 2 | 2 | 7 |
| $d_i$ | 2 | 3 | 4 | 8 |

Cuadro 1.1: Instancia para $1 \,|\, r_i; pmtn \,|\, L_{\text{máx}}$

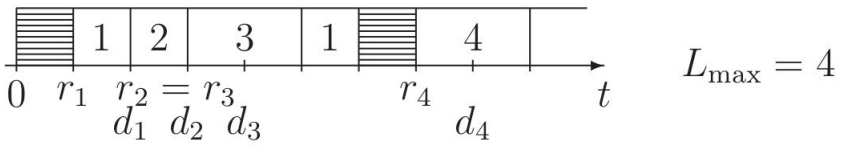


Figura 1.2: Horario factible para $1 \,|\, r_i; pmtn \,|\, L_{\text{máx}}$

## 1.2. Complejidad computacional

**Definición 1.6.** Sean $f$ y $g$ dos funciones, $f, g : \mathbb{N} \to \mathbb{R}$. Decimos que $f(n) \in \mathcal{O}(g(n))$ ($f(n)$ es de orden $g(n)$) si existe una constante $c \in \mathbb{R}^+$ y $n_0 \in \mathbb{N}$, tal que para cada número entero $n \geq n_0$, $f(n) \leq c\, g(n)$.

**Definición 1.7.** Definimos el **tiempo de ejecución de un algoritmo** como una función $T : \mathbb{N} \to \mathbb{N}$. Donde $T(n)$ es el número máximo de "pasos" en cualquier ejecución del algoritmo en las entradas de "tamaño" $n$.

**Observación 1.1.** Un paso del algoritmo se puede definir con precisión si se fija una máquina en particular en el que el algoritmo se va a ejecutar. Y para ser más precisos, para cada entrada $x$ se define el tamaño de la entrada $|x| = n$ como el número de bits necesarios para almacenar la entrada en alguna codificación de $x$.

**Definición 1.8.** Un algoritmo para un problema $P$ se dice que es de **tiempo polinomial** si existe un polinomio $p$, tal que $T(n) \in \mathcal{O}(p(n))$ para todas las entradas del problema, es decir, si existe un $k$ tal que $T(n) \in \mathcal{O}(n^k)$.

**Nota 1.1.** La noción de tiempo polinomial depende de la codificación. Se asume que todos los datos numéricos que describen el problema se codifican en binario.

**Definición 1.9.** Si el algoritmo es de tiempo polinomial cuando la entrada se da en la codificación unaria y de tiempo exponencial cuando la entrada se da en la codificación binaria, el algoritmo es llamado **algoritmo pseudopolinomial**.

**Nota 1.2.** En la codificación unaria todos los datos numéricos, que se suponen enteros, son codificadas por cadenas de unos. Más específicamente un número entero $d$ es representado por una secuencia de $d$ unos.

**Definición 1.10.** Un problema es llamado **problema de decisión** si el rango de salida es {si, no}.

**Nota 1.3.** A cada problema de programación de tareas se le puede asociar un problema de decisión, mediante la definición de un valor de cambio $k$ para la correspondiente función objetivo $\gamma$. Este problema de decisión es: ¿ existe un horario factible $\sigma$ tal que $\gamma(\sigma) = k$ ?.

**Definición 1.11.** La clase de todos los problemas de decisión que pueden ser resueltos en tiempo polinomial es denotada por $\mathcal{P}$.

**Definición 1.12.** La clase de problemas de decisión con la propiedad de que: para cualquier instancia para la cual la respuesta es " si ", existe un algoritmo de tiempo polinomial que lo verifica, es denotada por $\mathcal{NP}$.

**Definición 1.13.** Sean $P, Q$ dos problemas de decisión, si una instancia de $P$ se puede convertir en tiempo polinomial a una instancia de $Q$, entonces decimos que $P$ **es reducible polinomialmente** a $Q$.

**Observación 1.2.** El concepto de reducción es importante porque sirve para determinar la pertenencia de los problemas a las clases $\mathcal{P}$ y $\mathcal{NP}$, y permiten definir la clase $\mathcal{NP}$-completo.

**Definición 1.14.** La clase de problemas $\mathcal{NP}$-completo es el subconjunto de problemas $P \in \mathcal{NP}$ tal que para todo $Q \in \mathcal{NP}$, $Q$ es reducible polinomialmente a $P$.

**Definición 1.15.** Un problema de optimización se llama $\mathcal{NP}$-duro si el problema de decisión correspondiente es $\mathcal{NP}$-completo.

**Definición 1.16.** Un **algoritmo de $\rho$ -aproximación** para un problema de optimización es un algoritmo de tiempo polinomial tal que para todas las instancias del problema produce una solución factible cuyo valor es acotado por $\rho$ veces el valor de la solución óptima. $\rho$ se conoce como garantía de rendimiento o razón de rendimiento de la aproximación.

**Definición 1.17.** Una familia de algoritmos de aproximación de tiempo polinomial $A_\epsilon$ con garantía de rendimiento $1+\epsilon$, para todo $\epsilon > 0$ es llamado un **esquema de aproximación de tiempo polinomial (PTAS)**. Si el tiempo de ejecución de los algoritmos son además acotados por un polinomio en el tamaño de la entrada en $1/\epsilon$, entonces $A_\epsilon$ constituye un **esquema de aproximación totalmente de tiempo polinomial (FPTAS)**.

**Definición 1.18.** Considere los siguientes dos problemas de optimización:

$$(IP)\ z = \text{mín}\{c(x) : x \in X \subseteq \mathbb{Z}^n.\}$$
$$(RP)\ z^{RP} = \text{mín}\{f(x) : x \in T \subseteq \mathbb{R}^n\}.$$

El problema RP es llamado **relajación** de IP si $X \subset T$ y $f(x) \leq c(x), \forall x \in X$.

**Definición 1.19.** La **relajación de programación lineal** de $(IP)$ con formulación $P = \{x : Ax \leq b\}$ es el programa lineal $z^{LP} = \text{mín}\{c(x) : x \in P\}$.

## 1.3. El algoritmo del elipsoide

En esta sección se describe de forma general el algoritmo del elipsoide, del cual se hablara más adelante. El algoritmo del elipsoide para programación lineal es una aplicación específica del método del elipsoide desarrollado por los matemáticos soviéticos Shor, Yudin y Nemirovskii. Khachiyan utiliza el método del elipsoide para derivar el primer algoritmo de tiempo polinomial para la programación lineal. Aunque el algoritmo es teóricamente mejor que el método Simplex, que tiene un tiempo de ejecución exponencial en el peor de los casos, es muy lento y prácticamente no competitivo con

el método Simplex. Sin embargo, es una herramienta teórica muy importante para el desarrollo de algoritmos de tiempo polinomial para una amplia clase de problemas de optimización convexa, que son mucho más general que la programación lineal.

**Definición 1.20.** Un conjunto $K \subset \mathbb{R}^n$ es un conjunto **convexo** si $\forall k_1, k_2 \in K$ y $\lambda \in [0,1]$ el punto $\lambda x + (1-\lambda)y \in K$.

**Definición 1.21.** Un **oráculo de separación** para un conjunto convexo $K$ es un procedimiento mediante el cual dado un punto $p$, indica si $p \in K$ o devuelve un hiperplano que separa $p$ y todos los puntos de $K$. El procedimiento se ejecuta en tiempo polinomial.

El algoritmo del elipsoide funciona como sigue. Comienza con una gran elipsoide $E_0$ el cual garantiza que $K \subset E_0$. En cada paso del algoritmo pide el oráculo de separación alrededor del centro del elipsoide $p$. Si $p$ está en $K$ entonces la salida del algoritmo es el punto $p$. De lo contrario el algoritmo es capaz de dividir el espacio en dos (usando el hiperplano proporcionado por el oráculo de separación) y continuar la búsqueda en el lado correcto generando un nuevo elipsoide $E$. Se muestran dos iteraciones del algoritmo del elipsoide, para un caso particular en la figura.

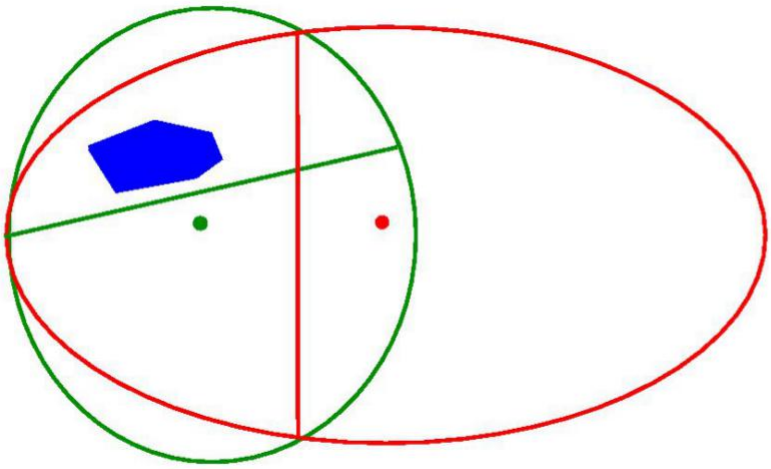

Figura 1.3: Ejemplo del algoritmo del elipsoide

# Capítulo 2

# Estudio de una aproximación para el problema

En este capítulo se describirá de manera detallada y precisa lo contenido en el artículo *"Approximation algorithms for scheduling problems with a modified total weighted tardiness objective"* [1], de los autores Stavros G. Kolliopoulos y George Steiner. Estos autores al observar la complejidad computacional del problema, optan por la posibilidad de algoritmos de aproximación, para así garantizar una solución factible próxima a la solución óptima. En este sentido logran dos resultados importantes, ya que se conocen pocos algoritmos de aproximación para problemas de programación de tareas.

## 2.1. Complejidad del problema

De todos los problemas de programación de tareas, hay una gran cantidad que son $\mathcal{NP}$-duro y por lo tanto no admiten algoritmos de tiempo polinomial que los resuelvan a optimalidad. Se procede a revisar brevemente lo que se conoce en la minimización de la tardanza ponderada total en una sola máquina, es decir sobre la minimización del problema denotado por $1 \mid\mid \sum_j w_j T_j$. Este problema fue clasificado como un problema $\mathcal{NP}$-duro, cuando las tareas sólo tienen dos fechas de entrega distintas, por una reducción del problema de la mochila [4]. Mucho más tarde, incluso para el caso de una sola fecha de entrega común se demostró que es un problema $\mathcal{NP}$-duro [5]. Lawler y Moore [6] han presentado una solución pseudopolinomial para este caso. Se conoce muy poco acerca de algoritmos de aproximación.

Curiosamente, la complejidad del problema con ponderación unitaria, es decir $1 \mid\mid \sum_j T_j$, fue un problema abierto por muchos años, hasta que se demostró ser $\mathcal{NP}$-duro [7]. Para

el caso $1\,|\,|\,\sum_j w_j T_j$, Kolliopoulos y Steiner [8] dieron un algoritmo pseudopolinomial , cuando sólo hay un número fijo de fechas de entrega diferentes. También han desarrollado un FPTAS si además, las ponderaciones $w_j$ son acotados por una función polinomial de $n$. Cheng, Ng, Yuan y Liu [9] han demostrado recientemente que el horario que minimiza máx $\{w_j T_j, j \in N\}$, logra una $(n-1)$-aproximación para $1\,|\,|\,\sum_j w_j T_j$. Esta es la única garantía de aproximación no trivial conocida para una versión del problema con los datos de entrada generales. No se tiene conocimiento de ninguna garantía de aproximación para problemas de programación de tareas con el objetivo de la tardanza ponderada total con varias máquinas.

## 2.2. Modificación del objetivo

Más adelante se presentaran un gran número de resultados de aproximación para problemas de programación de tareas de la forma $\alpha\,|\,\beta\,|\,\sum_j w_j(T_j + d_j)$. Es decir, los resultados se obtienen a partir de la modificación de la función objetivo original $(\sum_j w_j T_j)$. Es claro que para propósitos de optimización, este objetivo modificado es equivalente a minimizar la tardanza ponderada total, ya que agrega la constante independiente del horario $\sum_j w_j d_j$ a el objetivo. Se modifica el objetivo básicamente por dos razones, la primera es que parte de la dificultad de encontrar buenas aproximaciones para $\sum_j w_j T_j$, es que el óptimo puede ser cero, ya que si todas las tareas se completan a tiempo $T_j = 0, \forall j \in N$. Este tipo de irregularidades surgen para otros objetivos también, por ejemplo para la demora máxima $L_{\text{máx}}$. La manera habitual de hacer frente a esta dificultad es el uso de una formulación que añada una constante positiva a el objetivo. Para el caso de $L_{\text{máx}}$ la constante agregada transforma la demora de la tarea $j$ de $C_j - d_j$ a $C_j + q_j$, donde $q_j > 0$ es el denominado tiempo de entrega de la tarea $j$. Todos los resultados de aproximación conocidos para minimizar $L_{\text{máx}}$ son para este objetivo modificado.

Kovalyov y Werner [10] han estudiado la aproximación en el caso de ponderación unitaria en máquinas paralelas con fecha de entrega común, es decir para el problema $P_m\,|\,d_j = d\,|\,\sum_j T_j$. Usando el hecho de que el problema es $\mathcal{NP}$-completo para decidir si hay un horario con tardanza total cero, prueban que a menos que $\mathcal{P} = \mathcal{NP}$, no hay

un algoritmos de $\rho$-aproximación de tiempo polinomial para $P_m \,|\, d_j = d \,|\, \sum_j T_j$, con $\rho < \infty$. Esto muestra que las soluciones con valor cero para la función objetivo deben ser evitadas, para tener alguna esperanza de un factor de aproximación constante. Esto se puede hacer mediante la adición de una cantidad positiva apropiada a la función objetivo. El objetivo modificado mantiene la información sobre los tareas tardías. La segunda razón por la cual se modifica el objetivo se observa en la próxima sección.

## 2.3. Relaciones de aproximación

Los resultados obtenidos se basan en la explotación de la estrecha relación que hay entre los objetivos $\sum_j w_j C_j$ y $\sum_j w_j(T_j + d_j)$, en un gran número de entornos de programación. Se demuestra que la aproximación de $\sum_j w_j(T_j + d_j)$ se reduce a la aproximación de $\sum_j w_j C_j$, en el sentido de que cualquier algoritmo de $\rho$-aproximación para minimizar $\sum_j w_j C_j$ es un algoritmo de $(\rho + 1)$-aproximación para minimizar $\sum_j w_j(T_j + d_j)$.

### 2.3.1. Reducción a tiempo de completación total ponderado

**Teorema 2.1.** *Considere un miembro $\alpha_0 \,|\, \beta_0 \,|\, \sum_j w_j C_j$ de la familia de problemas de programación de tareas $\alpha \,|\, \beta \,|\, \sum_j w_j C_j$, para el cual existe un algoritmo de $\rho$-aproximación. Entonces el mismo algoritmo logra una $(\rho+1)$-aproximación para el problema $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$.*

*Demostración.* En primer lugar se probara la siguiente afirmación.

**Afirmación 2.1.** $T_j + d_j = \max\{C_j, d_j\}, \forall j \in N$.

*Demostración.* En efecto, sea $j \in N$ y $T_j$ la tardanza de la tarea $j$ en un horario arbitrario $\sigma$.
Si $T_j = 0$, entonces

$$\begin{aligned} \max\{C_j, d_j\} &= d_j \\ &= T_j + d_j. \end{aligned}$$

Si $T_j = C_j - d_j$, entonces

$$\begin{aligned} \max\{C_j, d_j\} &= C_j \\ &= (C_j - d_j) + d_j \\ &= T_j + d_j. \end{aligned}$$

■

Sea $\alpha_0 \,|\, \beta_0 \,|\, \sum_j w_j C_j$ un problema particular. Para cualquier horario $\sigma$, se cumple que:

$$\begin{aligned} \sum_j w_j C_j(\sigma) &\leq \sum_j w_j \max\{C_j(\sigma), d_j\} && (2.1) \\ &\leq \sum_j w_j C_j(\sigma) + \sum_j w_j d_j. && (2.2) \end{aligned}$$

$$\begin{aligned} OPT_{\sum_j w_j C_j} &\leq OPT_{\sum_j w_j \max\{C_j, d_j\}} && (2.3) \\ \sum_j w_j d_j &\leq OPT_{\sum_j w_j \max\{C_j, d_j\}}. && (2.4) \end{aligned}$$

Por hipótesis se tiene que existe un horario $\sigma^\rho$ que cumple con:

$$\sum_j w_j C_j(\sigma^\rho) \leq \rho OPT_{\sum_j w_j C_j}. \tag{2.5}$$

Por otro lado,

$$\begin{aligned} \sum_j w_j \max\{C_j(\sigma^\rho), d_j\} &\leq \sum_j w_j C_j(\sigma^\rho) + \sum_j w_j d_j, \text{ por (2.2)} \\ &\leq \rho OPT_{\sum_j w_j C_j} + \sum_j w_j d_j, \text{ por (2.5)} \\ &\leq \rho OPT_{\sum_j w_j \max\{C_j, d_j\}} + OPT_{\sum_j w_j \max\{C_j, d_j\}}, \text{ por (2.3) y (2.4)} \\ &\leq (\rho + 1) OPT_{\sum_j w_j \max\{C_j, d_j\}}. \end{aligned}$$

■

Se resumen las consecuencias más importantes del Teorema 2.1 en la siguiente tabla. Se omiten las relaciones que se mejoran más adelante con un enfoque basado en programación lineal.

| Problema | Relación para $\sum_j w_j(T_j + d_j)$ | Referencia para $\sum_j w_j C_j$ |
|---|---|---|
| $P \mid r_j, pmtn \mid \sum_j w_j(T_j + d_j)$ | | |
| $P \mid r_j \mid \sum_j w_j(T_j + d_j)$ | $2 + \epsilon$ | [11] |
| $R_m \mid r_j, pmtn \mid \sum_j w_j(T_j + d_j)$ | | |
| $R_m \mid r_j \mid \sum_j w_j(T_j + d_j)$ | | |
| $Q \mid pmtn \mid \sum_j (T_j + d_j)$ | 2 | [12] |
| $Q \mid r_j \mid \sum_j w_j(T_j + d_j)$ | $2 + \epsilon$ | [13] |

Cuadro 2.1: Relaciones de aproximación obtenidas por el teorema 2.1

### 2.3.2. Uso de relajación lineal

En esta sección se muestra que es posible mejorar aún más, las garantías de aproximación obtenidas en el teorema 2.1, en los casos en que algoritmos basados en programación lineal con ciertas características están disponibles para la aproximación del problema $\alpha \mid \beta \mid \sum_j w_j C_j$. Se propone una familia de relajaciones lineales para el objetivo modificado y se usa para demostrar que un horario con una relación de $\rho$-aproximación para el problema $\alpha \mid \beta \mid \sum_j w_j C_j$ es también un horario con la misma relación de aproximación para el problema $\alpha \mid \beta \mid \sum_j w_j(T_j + d_j)$ correspondiente.

La familia de programas matemáticos propuestos son parametrizados en base a un conjunto de restricciones $\mathcal{C}(C)$. En principio las restricciones en $\mathcal{C}(C)$ son generalmente convexas. En estas restricciones solamente aparecen las variables de tiempos de completación.

**Definición 2.1.** Un conjunto de restricciones $\mathcal{C}(C)$ es un **conjunto válido de restricciones de tiempos de completación** si los tiempos de completación $C_j$ $(j = 1, 2, \ldots, n)$ satisfacen las restricciones $\mathcal{C}(C)$, para todos los horarios factibles.

**Ejemplo 2.1.** Como un ejemplo concreto de $\mathcal{C}(C)$, considere el problema $1 \mid prec \mid \sum_j w_j(T_j + d_j)$, es decir, programación de tareas con restricciones de precedencia en una sola máquina. Un conjunto válido de restricciones de tiempos de completación, el cual se introdujo

en [14], es el siguiente:

$$C_k \geq C_j + p_k\,, \text{ para cada par } j,k \text{ tales que } j \prec k\,, \tag{2.6}$$

$$\sum_{j\in S} p_j C_j \geq \frac{1}{2}(p^2(S) + p(S)^2)\,,\ \forall S \subseteq N. \tag{2.7}$$

donde $p(S) = \sum_{j\in S} p_j$ , $p^2(S) = \sum_{j\in S} p_j^2$ y $j \prec k$ representa la restricción de precedencia. Queyranne ha demostrado que existe un oráculo de separación para el conjunto exponencialmente grande de restricciones (2.7) y por lo tanto se puede optimizar sobre él, en tiempo polinomial. Otros conjuntos válidos también se han dado en la literatura. Ver por ejemplo, los basados en variables de ordenación lineal [15] [16] [17] y la formulación indexada en el tiempo [18].

Sea $\mathcal{C}(C)$ un conjunto valido de restricciones de tiempos de completación para el problema $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$. Se utilizan formulaciones de programación matemática con variables $T_j, E_j$ y $C_j$. Se destaca que las variables $T_j$ o $E_j$ no tienen que aparecer en ninguna restricción de $\mathcal{C}(C)$. Las únicas variables involucradas pueden ser las variables de tiempo de completación y posiblemente, otras auxiliares. De hecho, en todas las formulaciones que se usan, las variables de tardanza y anticipación no aparecen en $\mathcal{C}(C)$. Se proponen la siguiente familia de programas lineales, que se denota $FP(\mathcal{C})$:

$$\begin{gathered} minimizar \sum_{j=1}^{n} w_j(T_j + d_j),\ sujeto\, a: \\ T_j = C_j - d_j + E_j, \quad j = 1, \ldots, n, \\ \mathcal{C}(C) \\ T_j, E_j, C_j \geq 0, \quad j = 1, \ldots, n. \end{gathered} \tag{2.8}$$

Dado que cada tarea es, ya sea anticipada o tardía en cualquier horario factible, es decir, a lo sumo una de las dos variables $T_j$ y $E_j$ puede ser positiva, es fácil ver que los valores $T_j$ y $E_j$ deben satisfacer las ecuaciónes (2.8) para cualquier horario factible. Por supuesto, las ecuaciones (2.8) permiten soluciones en las que ambas, tanto $T_j$ como $E_j$ son positivas, por lo tanto $FP(\mathcal{C})$ no es normalmente una formulación exacta para el problema $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$, es sólo una relajación de programación lineal.

Los resultados se basan en el siguiente esquema de algoritmo para el problema general $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$. Los valores de las variables de tiempo de completación regresados por una solución óptima de $FP(\mathcal{C})$, pueden no ser enteros y se refieren a estos valores como los tiempos de completación fraccionarios. El esquema supone la existencia de una subrutina $\mathcal{A}(\alpha, \beta)$ que encuentra un horario factible $\sigma$ que viene con una garantía de aproximación tarea-por-tarea para los tiempos de completación, es decir, si $\overline{C_j}$ es el tiempo de completación fraccionario y $C_j(\sigma)$ es el tiempo de completación en $\sigma$ para la tarea $j$, entonces se tiene que: $C_j(\sigma) \leq \rho\overline{C_j}$, $j = 1, \ldots, n$, para algún $\rho \geq 1$.

**ESQUEMA DE ALGORITMO**

1. Calcular una solución óptima para $FP(\mathcal{C})$. Sean $\overline{T_j}$, $\overline{E_j}$ y $\overline{C_j}$, los valores resultantes de las variables, $j = 1, \ldots, n$.

2. Al invocar un algoritmo apropiado $\mathcal{A}(\alpha, \beta)$, calcular un horario factible $\sigma$ para $\alpha \,|\, \beta \,|\, \sum_j w_j C_j$ en el cual $C_j \leq \rho\overline{C_j}, j = 1, \ldots, n$, para algún $\rho \geq 1$.

3. Salida del horario $\sigma$.

El análisis de los diferentes algoritmos depende del siguiente lema fundamental.

**Lema 2.1.** *Si el algoritmo $\mathcal{A}(\alpha, \beta)$ asumido en el paso 2 del ESQUEMA DE ALGORITMO existe, el horario salida $\sigma$ logra una $\rho$-aproximación para el problema $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$.*

*Demostración.* El horario $\sigma$ es factible para el ambiente de la máquina $\alpha$ y las características de las tareas $\beta$ por construcción. Para $j \in N$ se tiene que:

$$\begin{aligned} T_j(\sigma) &= C_j(\sigma) - d_j + E_j(\sigma) \\ &\leq \rho\overline{C_j} - d_j + E_j(\sigma) \end{aligned} \tag{2.9}$$

- Si la tarea $j$ no es tardía, $T_j(\sigma) = 0$ y la contribución de la tarea $j$ al objetivo es $w_j d_j$.

- Si la tarea $j$ es tardía, la contibución de $j$ al objetivo es $T_j(\sigma) + d_j$, que por (2.9) es a lo sumo:

$$T_j(\sigma) + d_j \leq w_j(\rho\overline{C_j} + E_j(\sigma)) = \rho\overline{C_j}. \tag{2.10}$$

Pero desde (2.8) se tiene que:

$$w_j(\overline{T_j} + d_j) = w_j(\overline{C_j} + \overline{E_j}).$$

Por lo tanto

$$w_j\overline{C_j} \leq w_j(\overline{T_j} + d_j). \tag{2.11}$$

Por otro lado,

$$\begin{aligned} w_j(T_j(\sigma) + d_j) &= w_j((C_j(\sigma) - d_j) + d_j) \\ &= w_j C_j(\sigma) \\ &\leq \rho w_j \overline{C_j}, \text{ por (2.10)} \\ &\leq \rho w_j(\overline{T_j} + d_j), \text{ por (2.11)}. \end{aligned}$$

■

Se resumen las consecuencias de lo expuesto hasta ahora en el siguiente teorema, el cual ya ha sido demostrado.

**Teorema 2.2.** *Si para un problema* $\alpha \,|\, \beta \,|\, \sum_j w_j(T_j + d_j)$

*(i) existe un conjunto válido de restricciones de tiempos de completación.*

*(ii)* $FP(\mathcal{C})$*, donde* $\mathcal{C}(C)$ *se sustituye por un conjunto válido especifico, puede ser resuelto en tiempo polinomial y*

*(iii) existe un algoritmo de* $\rho$*-aproximación para* $\sum_j w_j C_j$ *con la garantía tarea-por-tarea,*

*entonces hay un algoritmo de* $\rho$*-aproximación para* $\sum_j w_j(T_j + d_j)$*.*

*Demostración.* ■

Los principales problemas para los cuales se cumplen los requerimientos del teorema 2.2 se muestran en la siguiente tabla.

| Problema | Relación para $\sum_j w_j(T_j + d_j)$ | Referencia para $\sum_j w_j C_j$ |
|---|---|---|
| $1 \mid prec \mid \sum_j w_j(T_j + d_j)$ | 2 | [19] |
| $1 \mid r_j, prec, pmtn \mid \sum_j w_j(T_j + d_j)$ | 2 | [19] |
| $1 \mid r_j, prec \mid \sum_j w_j(T_j + d_j)$ | $e + \epsilon$ | [20] |
| $P \mid r_j, prec, pmtn \mid \sum_j w_j(T_j + d_j)$ | 3 | [19] |
| $P \mid r_j, prec \mid \sum_j w_j(T_j + d_j)$ | 4 | [21] |
| $Q \mid r_j, prec, pmtn \mid \sum_j w_j(T_j + d_j)$ | $\mathcal{O}(\log m)$ | [22] |
| $Q \mid r_j, prec \mid \sum_j w_j(T_j + d_j)$ | | |
| $R \mid \mid \sum_j w_j(T_j + d_j)$ | 1.5 | [23] |
| $R \mid r_j \mid \sum_j w_j(T_j + d_j)$ | 2 | [23] |
| $R \mid pmtn \mid \sum_j w_j(T_j + d_j)$ | 2 | [23] |
| $R \mid r_j, pmtn \mid \sum_j w_j(T_j + d_j)$ | 3 | [23] |

Cuadro 2.2: Relaciones de aproximación obtenidas por el teorema 2.2

## 2.4. Programación de tareas en una sola máquina con restricciones de precedencia

En esta sección se muestra un algoritmo de 2-aproximación para $1 \mid prec \mid \sum_j w_j C_j$ con la garantía tarea-por-tarea, dado en [19]. Dicho algoritmo es basado en la formulación de programación lineal que minimiza $\sum_j w_j C_j$, sujeto a las restricciones del conjunto valido $\mathcal{C}(C)$ dado en el ejemplo 2.1. La clave para la calidad de la aproximación que deriva de esta relajación es el siguiente lema.

**Lema 2.2.** *Sea $C_1, \ldots, C_n$ satisfaciendo (2.7) y asuma sin pérdida de generalidad que $C_1 \leq \cdots \leq C_n$, entonces para cada tarea $j = 1, \ldots, n$, $C_j \geq \frac{1}{2} \sum_{k=1}^{j} p_k$.*

*Demostración.* La desigualdad 2.7 para $S = \{1, \ldots, j\}$ implica que:

$$\begin{aligned} \sum_{k=1}^{j} p_k C_k &\geq \frac{1}{2}(p^2(S) + p(S)^2) \\ &\geq \frac{1}{2} p(S)^2. \end{aligned} \tag{2.12}$$

Ya que $C_k \leq C_j$, para cada $k = 1, \ldots, j$ tenemos

$$\begin{aligned} C_j p(S) &= C_j \sum_{k=1}^{j} p_k \\ &\geq \sum_{k=1}^{j} p_k C_k \\ &\geq \frac{1}{2} p(S)^2. \end{aligned}$$

o equivalentemente, $C_j \geq \frac{1}{2} \sum_{k=1}^{j} p_k$. ■

Considere la siguiente heurística para la producción del horario:

1. Se obtiene una solución óptima del problema de programación lineal.
2. Sean $\overline{C_1}, \ldots, \overline{C_n}$ los valores óptimos, las tareas se programan en función de los tiempos de completación fraccionarios, en orden no decreciente, es decir, la primera tarea a procesar en el horario es la que tiene menor tiempo de completación fraccionario $\overline{C_j}$.

Nos referimos a este algoritmo como horario-por-$\overline{C_j}$. Observe que las restricciones (2.6) aseguran que el horario resultante sea consistente con las restricciones de precedencia.

Queyranne en [14] ha demostrado que el programa lineal que minimiza $\sum_j w_j C_j$ es resoluble en tiempo polinomial a través del algoritmo del elipsoide. La observación clave es que hay un algoritmo de separación de tiempo polinomial para el conjunto exponencialmente grande de restricciones (2.7).

**Teorema 2.3.** *Sean $C_1^*, \ldots, C_n^*$ los tiempos de completación en algún horario óptimo y $\tilde{C}_1, \ldots, \tilde{C}_n$ los tiempos de completación en el horario encontrado por el algoritmo horario-por-$\overline{C_j}$, entonces $\sum_j w_j \tilde{C}_j \leq 2 \sum_j w_j C_j^*$. Es decir, el algoritmo horario-por-$\overline{C_j}$ es un algoritmo de 2-aproximación para $1 \,|\, prec| \sum_j w_j C_j$.*

*Demostración.* Por simplicidad asumimos que las tareas se han vuelto a numerar de modo que $\overline{C_1} \leq \cdots \leq \overline{C_n}$, por lo tanto, para $S = \{1, \ldots, j\}$, $\tilde{C}_j = \sum_{k=1}^{j} p_k$.
Por el lema 2.2 se obtiene que:

$$\overline{C_j} \geq \frac{1}{2} \sum_{k=1}^{j} p_k = 1/2\, \tilde{C}_j,$$

de donde ganamos la garantía tarea-por-tarea:

$$\tilde{C}_j \leq 2\,\overline{C_j}.$$

Puesto que $w_j \geq 0$ para cada $j = 1, \ldots, n$ y $\sum_j w_j \overline{C_j} \leq \sum_j w_j C_j^*$ se obtiene que:

$$\sum_j w_j \tilde{C}_j \leq 2 \sum_j w_j C_j^*.$$

■

Ahora bien, usando este algoritmo con los tiempos de completación $\overline{C_j}$ fraccionarios resultantes de una solución óptima de $FP(\mathcal{C})$, obtenemos un algoritmo de 2-aproximación para $1\,|\,prec|\,\sum_j (T_j + d_j)$.

# CONCLUSIONES

Se expone una pequeña lista de conclusiones, las primeras relativas al paper que sirvió de apoyo al T.E.G. y las ultimas al T.E.G. en sí. Se enumeran de esa manera para enfatizar la importancia de este T.E.G. .

1. El tema tratado en este T.E.G. es de suma importancia dentro de la sociedad actual.

2. El potencial de mejora es evidente ante lo descrito en el ítem anterior, debido a que algunos empresarios venezolanos actúan según su instinto, dando un potencial grande a estas herramientas.

3. El aporte teórico de la publicación de Kolliopolus y Steiner es contundente.

4. El aporte teórico es amplio.

5. El aporte teórico es formal, tiene un rigor matemáco incuestionable.

6. El aporte principal de este T.E.G. consiste en aclarar lo expuesto por Kolliopolus y Steiner.

7. Otro aporte importante del T.E.G. lo constituye la base conceptual capturada desde otras publicaciones y la exposición justa y adecuada dentro del contexto del tema.

Entre las recomendaciones tenemos:

- Continuar con la línea de investigación mediante otros T.E.G. y tesis de maestría del decanato.

- Discutir la posibilidad de insertar contenidos de programación de tareas en el pensum de la carrera Ingeniería de Producción.

# Bibliografía


[1] George Steiner Stavros G. Kolliopoulos. Approximation algorithms for scheduling problems with a modified total weighted tardiness objective. *Operations Research Letters*, 35:685–692, 2007.

[2] Peter Brucker. *Scheduling Algorithms*. Springer Berling Heidelberg, New York, $5^{th}$ edition, October 2006.

[3] J. K. Lenstra A. H. G. R. Kan R. L. Graham, E. L. Lawler. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[4] P. Brucker J. Lenstra, A. R. Kan. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[5] J. Yuan. The np-hardness of the single machine common due date weighted tardiness problem. *Systems Science and Mathematical Sciences*, 5:328–333, 1992.

[6] J. M. Moore E. L. Lawler. A functional equation and its application to resource allocation and sequencing problems. *Management Science*, 6:77–84, 1969.

[7] J. Y.T. Leung J. Du. Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 15:483–495, 1990.

[8] G. Steiner S. G. Kolliopoulos. Approximation algorithms for minimizing the total weighted tardiness on a single machine. *Theoretical Computer Science*, 355:261–273, 2006.

[9] J. J. Yuan Z. H. Liu T. C. E. Cheng, C. T. Ng. Single machine scheduling to minimize total weighted tardiness. *European Journal of Operational Research*, 165:423–443, 2005.

[10] F. Werner M. Y. Kovalyov. Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness. *Journal of Heuristics*, 8:415–428, 2002.

[11] C. Chekuri D. Karger C. Kenyon S. Khanna I. Milis M. Queyranne M. Skutella C. Stein M. Sviridenko F. Afrati, E. Bampis. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science*, 1999.

[12] T. Gonzalez. Optimal mean finish time preemptive schedules. Tech. rep. tech. rep. 220, comp. sci. dept., Pennsylvania State University, 1977.

[13] S. Khanna C. Chekuri. A ptas for minimizing weighted completion time on uniformly related machines. In *Proceedings of the 28th ICALP*, 2001.

[14] M. Queyranne. Structure of a simple scheduling polyhedron. *Mathematical Programming*, 58:263–285, 1993.

[15] C. N. Potts. An algorithm for the single machine sequencing problem with precedence constraints. *Mathematical Programming Study*, 13:78–87, 1980.

[16] D. S. Hochbaum F. A. Chudak. A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. *Operations Research Letters*, 25:199–204, 1999.

[17] Y. Wang F. Margot, M. Queyranne. Decompositions, network flows and a precedence constrained singlemachine scheduling problem. *Operations Research*, 51:981–992, 2003.

[18] L. Wolsey M. Dyer. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Applied Mathematics*, 26:255–270, 1990.

[19] D. B. Shmoys J. Wein L. A. Hall, A. Schulz. Scheduling to minimize average completion time: Offline and on-line approximation algorithms. *Mathematics of Operations Research*, 22:513–544, 1997.

[20] M. Skutella A. Schulz. Random-based scheduling: New approximations and lp lower bounds. In *Proc. 1st Int. Symp. on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 1997.

[21] A. Schulz A. Munier, M. Queyranne. Approximation bounds for a general class of precedence constrained parallel machine scheduling problems. In *Proceedings of the 6th Conference on Integer Programming and Combinatorial Optimization*, 1998.

[22] D. B. Shmoys F. A. Chudak. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30:323–343, 1999.

[23] M. Skutella. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48:206–242, 2001.