VIII Congreso de Ingeniería de Organización
Leganés, 9 y 10 de septiembre de 2004

Problemas de programación de tareas en máquinas en paralelo

Liliana Capacho Betancourt[1], Rafael Pastor Moreno[2]

[1] Departamento de Investigación de Operaciones, Escuela de Ingeniería de Sistemas y Centro de simulación y modelado CESIMO – Universidad de Los Andes, Mérida, Venezuela. Instituto de Organización y Control de Sistemas Industriales, Universidad Politécnica de Cataluña, Barcelona, España. liliana.capacho@upc.es
[2] Instituto de Organización y Control de Sistemas Industriales, Barcelona, España. rafael.pastor@upc.es

Resumen

En este trabajo se presenta un análisis de los enfoques usados para resolver problemas de programación de tareas en máquinas en paralelo, haciendo énfasis en los problemas en los que se considera la asignación de recursos limitados. El problema general de programación de tareas contempla la asignación y secuenciación de un conjunto de n tareas que deben ser procesadas en un conjunto de m máquinas. En el caso de máquinas en paralelo, los problemas de programación de tareas presentan un alto nivel de complejidad y son muy difíciles de resolver, por lo que en general se estudian casos muy simples cuyos supuestos consideran: máquinas idénticas que pueden procesar un número arbitrario de tareas, máquinas que pueden procesar una sola tarea a la vez, no se permiten interrupciones, no hay tiempos de setup y si los hay son independientes de la secuencia de las tareas, los tiempos de proceso son fijos, conocidos a priori e independientes de la secuencia, no hay restricciones de precedencia y las funciones de optimización son mono-objetivo. En los problemas más complejos se varía uno (o más) de dichos supuestos.

Palabras clave: programación, secuenciación, máquinas en paralelo.

1. Introducción

El problema general de asignación y secuenciación contempla un conjunto de *n* tareas que deben ser procesadas en un conjunto de *m* máquinas; y se considera que cada tarea se realiza en una sola máquina y que cada máquina procesa una sola tarea a la vez. Dichos problemas están caracterizados principalmente por los tiempos de proceso de cada tarea, la diferencia o igualdad de las máquinas (es decir, si todas las máquinas ejecutan las mismas tareas o están especializadas en cierto tipo de tareas) y por las diversas restricciones impuestas en la realización de las tareas. Los algoritmos de asignación y secuenciación consideran un criterio de eficiencia que se establece en la función objetivo.

Błażewicz et al. (1996) caracterizan a los problemas de asignación y secuenciación de tareas con tres conjuntos: un conjunto $T = \{T_1, T_2, ..., T_n\}$ de n tareas, un conjunto $P = \{P_1, P_2, ..., P_m\}$ de m máquinas y un conjunto $R = \{R_1, R_2, ..., R_S\}$ de s tipos adicionales de recursos. De esta forma, la secuenciación implica asignar las máquinas del conjunto *P*, y (posiblemente) los recursos del conjunto *R*, a las tareas del conjunto *T*, de forma que se completen todas las tareas, dadas ciertas restricciones. Para clasificar los problemas de este tipo, Błażewicz et al. (1996) utilizan la notación propuesta por Graham et al. (1979) compuesta por tres campos **α|β|γ**, en donde: $\boldsymbol{\alpha}=\alpha_1,\alpha_2$ describe el ambiente de la máquina (α_1 caracteriza el tipo de máquina y α_2 especifica el número de máquinas usadas), **β** describe las tareas y las características de los

recursos y γ denota el criterio de optimización o la medida de desempeño usada.

Los parámetros α, β y γ pueden tomar los siguientes valores:

- $\alpha_1 = \{\varnothing, P, Q, R, O, F, J\}$. Si $\alpha_1 = \varnothing$ se está en el caso particular de una sola máquina; $\alpha_1 = P$ significa que las máquinas son idénticas, es decir, el tiempo para procesar una tarea *j* es el mismo en cualquier máquina *i* ($p_{ij} = p_j$); $\alpha_1 = Q$ especifica máquinas uniformes o proporcionales, es decir, el tiempo de proceso de una tarea *j* depende de la velocidad constante, s_i, de la máquina *i* ($p_{ij} = p_j / s_i$); si $\alpha_1 = R$ se contemplan máquinas no relacionadas, en este caso la velocidad de la máquina depende de la tarea particular que se este procesando y puede ser diferente para cada máquina. Los diferentes tipos de flujo se especifican con: *O* para *open shop, F* para *flow shop* y *J* para *job shop*.

- $\alpha_2 \in \{\varnothing, k\}$: si $\alpha_2 = \varnothing$ el número de máquinas es variable; si $\alpha_2 = k$ el problema contempla *k* máquinas.

- $\beta = \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$ permite especificar: $\beta_1 = \{\varnothing, pmtn\}$ si se permiten o no interrupciones; $\beta_2 = \{\varnothing, res\}$ si hay recursos adicionales; β_3 si hay restricciones de precedencia; $\beta_4 = \{\varnothing, r_j\}$ indica cuándo las tareas están disponibles (*ready times*); $\beta_5 = \{\varnothing, p_j\}$ especifica los tiempos de proceso; β_6 define las fechas de entrega (*deadlines*); β_7 describe el número máximo de tareas que constituyen un trabajo en el caso de problemas tipo *job shops*; y $\beta_8 = \{\varnothing, no\_wait\}$ indica si una tarea, una vez procesada en una máquina, debe ser procesada inmediatamente en la máquina siguiente.

- El tercer campo, $\gamma \in \{C_{\max}, \sum C_j, \sum w_j C_j, L_{max}, \sum D_j, \sum w_j D_j, \sum E_j, \sum w_j E_j, \sum w_j U, -\}$, indica el objetivo de la programación de las tareas, y puede ser, por ejemplo, minimizar C_{max} (tiempo máximo para procesar una tarea), L_{max} (retraso máximo de las tareas), etc. Otros objetivos comúnmente usados incluyen: maximizar la utilización de los recursos del sistema, maximizar la tasa de producción, balancear el uso de los recursos y minimizar el inventario en progreso.

Por ejemplo, $P \mid pmtm \mid C\max$ representa un problema de máquinas paralelas idénticas en el que se permiten interrupciones, con el objetivo de minimizar C_{max}.

Los problemas de asignación y secuenciación en dos o más máquinas en paralelo han mostrado ser muy complejos: Błażewicz et al. (1996) concluyen que el problema es difícil de resolver aún para problemas simples como el caso de dos máquinas, $P2 \mid \mid C_{\max}$, para el cual se ha probado que es NP-*hard*.

Es obvio que la imposición de restricciones sobre problemas de este tipo, como por ejemplo la dependencia de los tiempos de proceso a la asignación de recursos, complica aun más su resolución. Dada la dificultad de tratamiento de este tipo de problema, la mayoría de trabajos abordan casos más simples.

En la sección 2 se presentan los tipos de problemas de programación de máquinas en paralelo tratados en la literatura y se expone el caso particular de asignación de recursos.

2. Problemas de asignación y secuenciación de tareas en máquinas en paralelo

Como se ha mencionado, dada su dificultad de resolución, la mayor parte de los trabajos expuestos en la literatura tratan problemas sencillos, en los que, por ejemplo, se consideran máquinas idénticas que procesan un número arbitrario de tareas, no hay restricciones de asignación de las tareas a máquinas particulares, todas las máquinas pueden procesar cualquier tarea y se contemplan funciones mono-objetivo. A continuación se presentan algunos casos menos restrictivos.

2.1. Asignación y secuenciación de tareas con fechas de entrega comunes

De acuerdo con Gordon et al. (2002), los problemas de programación de tareas con fechas de entrega comunes (*common due dates*) han recibido especial atención en los últimos 15 años, debido a la introducción de nuevos enfoques de administración de inventarios tales como el enfoque justo-a-tiempo (*just-in-time*), en donde el objetivo es completar las tareas lo más cerca posible de su fecha de entrega. Gordon et al. (2002) presentan un estado del arte de los problemas de asignación y secuenciación de tareas en una máquina simple y en máquinas en paralelo, en donde la fecha de entrega común es una variable de decisión. Funciones objetivos habitualmente consideradas en la literatura incluyen: minimizar la desviación absoluta media (MAD) y la suma ponderada de la desviación de los tiempos de entrega y los tiempos de finalización de las tareas (WSAD), así como minimizar el peso ponderado de *earliness*/*tardiness* (WET). Gordon et al. (2002) comentan la complejidad computacional que caracteriza a los diversos tipos de problemas que consideran fechas de entrega comunes: gran parte de dichos problemas son NP-*hard*.

Brucker (1998) analiza problemas de asignación y secuenciación de tareas en máquinas en paralelo que involucran fechas de entregas comunes y presenta la complejidad computacional de los algoritmos utilizados para resolverlos.

Sun et al. (2003) estudian el caso particular WET en donde se requiere programar un conjunto de tareas con fecha de entrega comunes en máquinas en paralelo idénticas, penalizando *earliness* y *tardiness*, con un peso de cada tarea proporcional a su tiempo de proceso. Sun et al. (2003) proponen un algoritmo de programación dinámica y dos heurísticas para resolver este tipo de problemas. Las heurísticas están basadas en el procedimiento de listas, en donde la próxima tarea de la lista se programa en la primera máquina que este disponible; dada la asignación de las tareas se requiere determinar el tiempo de comienzo de la primera tarea en cada máquina.

Finke et al. (2002) asumen que cada tarea tiene su propia fecha de entrega y que tienen penalizaciones de *earliness* y *tardiness* comunes. El tiempo de proceso de cada tarea es determinístico y conocido por adelantado, así como la asignación de las tareas a las máquinas; todos los trabajos son conocidos al inicio; no se permiten interrupciones; y los tiempos de preparación (*setup*) están incluidos en el tiempo de proceso. Utilizan *tabu search* y simulación para resolver el problema.

2.2. Asignación y secuenciación de tareas con tiempos de proceso estocástico

En este caso, los tiempos de proceso de las tareas no son fijos y conocidos a priori sino son variables aleatorias. Un caso especial de este tipo de problemas considera tiempos de proceso exponenciales. Pinedo (1995) analiza tres tipos de modelos para el caso de máquinas en paralelo: sin interrupciones con el objetivo de minimizar Cmax, con interrupciones considerando además el objetivo de minimizar el tiempo total de completar las tareas y modelos que contemplan fechas de entrega comunes.

2.3. Programación de tareas con funciones multi-objetivo

Tamaki et al. (1996) consideran un problema de programación de tareas que trata el caso de máquinas en paralelo idénticas con dos tipos de función objetivo. Una función objetivo regular, que trata de minimizar la diferencia máxima entre el tiempo de finalización de la última tarea Z_i^E procesada y el tiempo de inicio de la primera Z_i^s ($f_1=max_i(Z_i^E-Z_i^s)$). Y una segunda función objetivo trata de minimizar la suma ponderada de los retrasos y adelantos: ($f_2=\sum(\alpha E_i + \beta T_i)$).

Tamaki et al. (1996) descomponen el problema en tres subproblemas: uno para decidir la asignación de las tareas a las máquinas, uno para secuenciar las tareas en cada máquina y otro para decidir los tiempos de comienzo de cada tarea; y proponen un algoritmo genético para secuenciar y asignar los tareas a las máquinas, utilizando un modelo de programación lineal para determinar los tiempos de comienzo de cada tarea.

2.4. Problemas con tareas de multi-operación

En algunos casos se considera que las tareas están divididas en *r* partes que deben ejecutarse en diferentes máquinas con tiempos de proceso que se solapan, por lo que cada tarea es procesada en r máquinas de forma simultánea.

En Lee et al. (1997) pueden encontrarse descritos algunos trabajos de este tipo. También, se puede consultar el artículo de Hou y Hoogeveen (2003) en el que se estudia un problema de tareas procesadas en *r* máquinas pero con velocidades diferentes.

Por su parte, Mao (1997) define un problema de asignación y secuenciación de tareas en máquinas en paralelo, en donde la ejecución de las tareas incluye múltiples operaciones a ser realizadas por máquinas de diferentes tipos; por tanto, contempla restricciones acerca del orden en el cual las máquinas serán usadas para ejecutar las tareas. El autor comenta que este tipo de problemas se originan en manufactura industrial, planificación de la producción y en control automatizado. Además, prueba que este problema es *NP-hard complete*, por lo que diseña y analiza dos heurísticas tipo *greedy* para resolverlo con el objetivo de minimizar el *makespan*.

2.5. Programación de tareas en máquinas multi-propósito

Brucker et al. (1997) estudian la asignación y secuenciación en máquinas en paralelo que son multi-propósito, es decir, una máquina puede realizar varios tipos de tareas; en este caso una tarea u operación puede ser procesada por cualquier máquina de un subconjunto asociado, especificado a priori, del conjunto de máquinas disponibles.

El caso general considera que cada máquina procesa un único tipo de tarea. Para describir un problema de asignación y secuenciación con máquinas multi-propósito, y siguiendo la notación previamente descrita, se agrega MPM después del campo α: αMPM|β|γ. En Brucker et al. (1997) se presenta la complejidad computacional de este tipo de problemas; se argumenta que el problema R MPM||$\sum C_i$ es equivalente al problema R||$\sum C_i$, y que éste se puede formular como un problema de encuentro ponderado (*weighted matching*) que puede resolverse en tiempo $O(n^3)$; y que todos los otros problemas con tiempo de proceso no conocidos a priori y máquinas en paralelo multi-propósito son NP-*hard*.

2.6. Problemas con restricciones especiales sobre las máquinas

2.6.1. De capacidad

Existen algunos modelos que consideran restricciones de capacidad de las máquinas. En este caso, se considera a las máquinas como recursos limitados en las que únicamente se pueden procesar cierta cantidad de tareas. Yang et al. (2000) analizan un problema con restricciones de capacidad de las máquinas, en donde se debe secuenciar *n* tareas independientes en dos máquinas en paralelo idénticas, considerando un límite en el número de tareas que pueden ser asignadas a cada máquina, no se permiten interrupciones y el objetivo es minimizar *Cmax*.

2.6.2. De setup

También se han estudiado problemas en donde se tienen restricciones de operador (*operator constrained*) cuando se requieren cambios en la configuración de las máquinas entre una tarea (o grupo de tareas) y otra. Un ejemplo de este caso puede encontrarse en Herrmann et al. (1994), en donde un operador es el que debe configurar las máquinas para poder realizar las tareas; el problema consiste en secuenciar *n* tareas, y los *setups*, en *m* máquinas idénticas con el fin de minimizar el *Cmax,* sólo hay un operador y las tareas deben ejecutarse inmediatamente después del *setup.*

2.6.3. De mantenimiento

Este tipo de problema es conocido como *scheduling with machine availability constraints*. Lee y Chen (1998) consideran el caso en donde las máquinas necesitan recibir mantenimiento una vez durante el horizonte de planificación. La consecuencia inmediata es que ninguna tarea puede ser asignada a una máquina mientras permanezca en estado de mantenimiento. Estos autores consideran dos casos: uno, caso independiente, $Pm|ind|\sum w_j C_j$, en donde la disponibilidad de recursos permite que varias máquinas reciban mantenimiento a las vez: y otro, caso dependiente, $Pm|dep|\sum w_j C_j$, en donde el recurso de mantenimiento es único por lo que sólo una máquina puede estar en mantenimiento a la vez.

Para resolver este problema proponen algoritmos de *branch&bound* basados en el enfoque de generación de columnas y resuelven óptimamente problemas de tamaño medio. El modelo contempla programar simultáneamente la ejecución de las tareas y el mantenimiento de las máquinas. El proceso de una tarea no se puede reiniciar, esto es, si se está procesando una tarea en el momento que le corresponde mantenimiento a la máquina, la tarea deberá procesarse de nuevo completamente. Las tareas tienen un tiempo de proceso P_j y la función objetivo considera minimizar tiempos ponderados de culminación de las tareas.

De acuerdo con Lee y Chen (1998), se ha realizado poca investigación para modelar de manera conjunta la programación de tareas y la programación de las actividades de mantenimiento.

2.7. Programación de tareas en máquinas en paralelo con asignación de recursos

Como se ha mencionado, en la mayoría de problemas de asignación y secuenciación de tareas se considera tiempos de proceso de las tareas fijos y conocidos a priori. Sin embargo, dado que la estrategia de asignar recursos adicionales a los procesos productivos es frecuentemente usada para mejorar su desempeño, ya que así se acelera la ejecución de las tareas, cada vez se da mayor atención a problemas en los que los tiempos de proceso no son fijos sino variables de decisión y función de la cantidad de recursos asignados. Daniels et al. (1994) consideran que la eficiencia de los procesos de manufactura puede ser mejorada significativamente si se coordina la asignación de recursos adicionales, requeridos para ejecutar una tarea, junto con el proceso de asignación y secuenciación de las tareas en las máquinas; en su trabajo contemplan el modelo general de *flow shop* reflejando la flexibilidad de recursos, es decir, determinando simultáneamente la secuencia de las tareas, los inicios de operación y las políticas de asignación de los recursos.

El problema de programación de tareas y asignación de recursos en máquinas en paralelo ha sido abordado de diversas maneras.

Daniels et al. (1996) analizan un problema en el cual se tienen que asignar y secuenciar un conjunto de tareas sobre un conjunto de celdas de manufactura paralelas. Cada celda consta de una máquina simple, en donde el tiempo de proceso de cada tarea depende de la cantidad de recurso asignado a la celda asociada. Se asume que hay una cierta cantidad de un recurso simple que debe ser compartido entre las m celdas de manufactura.

Daniels et al. (1996) presentan las formulaciones de algunas heurísticas diseñadas para tratar problemas de asignación y secuenciación de máquinas en paralelo con flexibilidad de recursos (PMCRF), en los que el tiempo de proceso de las tareas es dependiente de la posición de la tarea en la secuencia. Para el caso de celdas flexibles formulan dos versiones del problema: una estática, en donde el recurso es distribuido entre las máquinas en t=0 y esta asignación permanece fija durante todo el horizonte de planificación y una dinámica en donde el recurso puede ser dinámicamente reasignado a la tarea o a la celda j; además, no se permiten interrupciones por lo que la cantidad de recurso asignado a una tarea permanece comprometida mientras la tarea este siendo procesada.

Daniels et al. (1996) formulan la versión estática del problema como un programa lineal entero 0-1, y lo resuelven usando un procedimiento heurístico: inicialmente asignan la mínima cantidad de recurso a cada celda de manera que cada tarea pueda ser procesada en su modo más lento; a continuación, identifican la celda cuya carga de proceso total corresponde a la

duración (*makespan*) actual del proyecto y determinan la mínima cantidad de recurso adicional que debe ser asignado a esa celda para hacer que al menos una de las tareas pueda ser procesada en su siguiente modo más rápido; si queda aun suficiente recurso, se asigna recurso adicional a la celda identificada. El proceso se repite hasta que la mínima cantidad de recurso adicional requerido para reducir la duración del proyecto excede la cantidad remanente o hasta que todas las tareas en la celda crítica hayan sido asignadas en su modo de proceso más rápido.

Para el caso dinámico, desarrollan un algoritmo *branch&bound* con el que se enumerarían todas las políticas de asignaciones de recursos posibles (los nodos del procedimiento de búsqueda representan programas de recursos factibles), y luego definen un procedimiento que resuelve repetidamente series de subproblemas estáticos para determinar la secuencia de las tareas en cada celda; las secuencias son entonces usadas para generar la solución completa del caso dinámico. Con el método propuesto únicamente resuelven problemas muy pequeños.

Ólafsson y Shi (2000) proponen un método alternativo para resolver este problema: reformular el problema PMCRF propuesto por Daniels et al. (1996) utilizando particiones anidadas (*nested partitions*) en lugar de *branch&bound*; de esta forma solo se exploran un número de ramas limitado en cada iteración.

Zhi-Long (2001) también analiza este tipo de problemas y considera dos criterios de minimización del coste total de los recursos asignados: uno es el coste total de los tiempos de culminación ponderados de las tareas y el otro es el coste total de las tareas retrasadas. Utilizando la notación presentada, los casos contemplados por Zhi-Long (2001) son: $P|res|\sum w_j C_j + TCR$, $P|res|\sum w_j U_j + TCR$, en donde: $TCR = \sum x_j$ y x_j es el coste de asignar el correspondiente recurso a la tarea *j*; U_j es una variable binaria que vale 1 si la tarea esta retrasada y w_j es un peso asignado externamente. Para resolver estos problemas desarrolla algoritmos *branch&bound* basados en generación de columnas. Según Zhi-Long, se ha demostrado que este método es el más efectivo computacionalmente para problemas de programación de tareas en varias máquinas en paralelo y argumenta, de acuerdo con los resultados obtenidos, que los algoritmos son capaces de resolver problemas de mediano tamaño, por ejemplo problemas de 40 tareas y cualquier número de máquinas.

Nowicki (1993) analiza el problema *flow-shop* en máquinas en paralelo: $F||C_{\max}$, con tiempo de proceso dependientes de la decisión de aumentar [o disminuir] la cantidad de recursos asignada a una máquina. Presenta un algoritmo de aproximación, en donde tanto los tiempos de proceso de las tareas como el orden de proceso son variables de decisión. Asume que el coste de procesar una tarea en cada máquina es una función lineal de su tiempo de proceso y el coste de programar todas las tareas. Utiliza el algoritmo de *Johnson* para secuenciar las tareas; no se permiten interrupciones; y se considera que todas las máquinas son idénticas.

Alidaee y Ahmadian (1996) consideran y describen el problema como Nowicki, pero con la diferencia de que las máquinas no son idénticas. Se estudian dos casos particulares: uno trata de minimizar el coste total de proceso más el tiempo de proceso total; y el otro trata de minimizar el costo total de proceso más los adelantos (*earliness*) y retrasos (*tardiness*) ponderados. Cada caso es reducido a un modelo de transporte y las funciones del costo de proceso y las de tiempo de proceso pueden ser no lineales.

Trick (1994) estudia el mismo tipo de problemas que Alidaee y Ahmadian (1996) con máquinas diferentes aunque los tiempos de proceso no dependen de la asignación de recursos adicionales, sino de las capacidades, en tiempos de proceso, de cada máquina. Cada tarea tiene un máximo y un mínimo de tiempo en el que puede ser procesada en cierta máquina y los tiempos de proceso totales en cada máquina no deben exceder su capacidad. Por cada cantidad de tiempo de máquina asignado a cada tarea se incurre en un coste, que permite optimizar la asignación de tiempos de procesos de las tareas.

Por su parte, Bar-Noy et al. (2001) analizan el problema de programación en el que se tiene un conjunto de tareas que compiten por un recurso renovable, como por ejemplo el problema de asignación de banda ancha para sesiones en redes de comunicaciones y la programación de las tareas en las máquinas (cada tarea utiliza cierta cantidad del recurso durante su tiempo de ejecución y luego lo libera cuando ésta ha sido completada). En este caso, el problema consiste en seleccionar un subconjunto factible de tareas a ejecutar, de forma que la cantidad de recurso asignada simultáneamente a dichas tareas no exceda la cantidad total de recurso disponible. Cada tarea puede admitir varias alternativas para su ejecución pero sólo una debe seleccionarse; cada alternativa consiste en el tiempo durante el cual la actividad tendrá lugar, la demanda de recurso si la actividad tiene lugar en ese intervalo y la ganancia obtenida por programar la actividad en dicho intervalo. De esta forma, el objetivo es encontrar un programa factible que especifique qué tareas son seleccionadas y cuándo son ejecutadas de forma que se maximice el beneficio obtenido. También se considera el objetivo de minimizar las perdidas de ganancias debidas a actividades no programadas.

Como puede observarse, la mayoría de los trabajos de programación de tareas que consideran simultáneamente la asignación de recursos, consideran al recurso limitado renovable, no permiten interrupciones de las tareas, no contemplan restricciones de precedencia entre tareas y consideran el caso de máquinas idénticas.

Recientemente, se han presentando diversos estudios que enfocan este problema con un nivel de complejidad mayor. Por ejemplo, Shabtay y Kaspi (2002) asumen que el recurso limitado común es no renovable, consideran el problema de optimización con y sin interrupciones, utilizan una función de consumo de recurso convexa y consideran restricciones de precedencia. En su artículo, Shabtay y Kaspi presentan los algoritmos usados para resolver los diferentes casos especiales enmarcados en dichas consideraciones. El objetivo es minimizar la suma de los tiempos de finalización de las tareas.

Referencias

Alidaee B., Ahmadian A. (1993). Two parallel machine sequencing problems involving controllable job processing times. European Journal of Operational Research, 70, 335-341.

Bar-Noy A., Bar-yehuda R., Freund A. y Naor, J. (2001).A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, Vol. 48, no. 5, pp. 1069-1090.

Błażewicz J., Ecker K., Pesch E., Schmidt G., Węglarz J. (1996). Scheduling Computer and Manufacturing Processes, Springer-Verlag.

Brucker P., Jurisch B., Krãmer A. (1997). Complexity of scheduling problems with multi-purpose machines. Annals of Operations Research 70, 57-73.

Bruker P. (1998). Scheduling Algorithms. Springer-Verlag.

Daniels R., Mazzola J. (1994). Flow shop scheduling with resource flexibility. Operations Research. 42, 504-522.

Daniels R., Hoopes B. y Mazzola J. (1996). Scheduling parallel manufacturing cells with

resource flexibility. *Management Science*. 42, 1260-1276.
Daniels R., Hoopes B., Mazzola J. (1997). An analysis of heuristics for parallel-machine flexible-resource scheduling problems. Annals of Operations Research 70, 439-472
Finke D.A., Medeiros D.J., Traband M.T. (2002). Shop Scheduling using tabu search and simulation. Proceeding of the 2002 Winter Simulation Conference.
Gordon V., Proth G.M. y Chu C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Op. Research*. 139, 1-25.
Graham R.L., Lawler E.L., Johnson D.S., Lenstra J.K. y Rinnooy Kan. (1979). Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Ann. Discrete Maths, 5, 287-326.
Herrmann J., Lee C.-Y. (1994). On parallel-machine scheduling with operator-constrained setups. Technical Research report. T.R 94-85. *Institute for systems research*.
Hou S., Hoogeveen H. (2003). The three-machine proportionate flow shop problems with unequal machine speeds. Operations research letters 31, 225-231.
Lee C., Lei L. y Pinedo M. (1997). Current trends in deterministic scheduling. Annals of operations Research 70, -41.
Lee C.Y., Chen Z.L. (1998). Scheduling Jobs and maintenance activities on parallel machines. Naval Research Logistics.
Mao Weizhen. (1997). A Parallel Multi-Operation Schedulling Problem with Machine Order Constraints. Pgs. 473-477. ACM.
Nowicki E. (1993). An approximation algorithm for the m-machine permutation flow shop problem with controllable processing times. *European Journal of Op. Research*, 70 342-349.
Ólafsson S. y Shi L. (2000). A methods for scheduling in parallel manufacturing systems with flexible resources. *IEE Transactions*. 32, 135-146.
Pinedo M. (1995). SCHEDULING Theory, Algorithms, and Systems.Prentice Hall.
Shabtay D., Kaspi M. (2002). Parallel machine scheduling with a convex resource consumption function. (Por publicar)
Sun H., Wang G. (2003). Parallel machine earliness and tardiness scheduling with proportional weights. Computers&Operational Research 30, 801-808.
Tamaki H. (1996). Nishino E., Abe S. A Genetic algorithm Approach to Multi-Objective Schedulling Problems with Earliness and Tardiness Penalties. IEEE, pgs. 46-52.
Trick M.A. (1994). Scheduling multiple variable-speed machines. Oper. Res. 42, 234-248.
Vickson R.G. (1980). Choosing the Jobs sequencing and processing times to minimize total processing plus and flow cost on a single machine. Oper. Res., Vol.28, No.5. 1155-1167.
Zhi-Long C. (2001). Simultaneous Job Scheduling and Resource Allocation on Parallel Machines. http://bmgt3-notes.umd.edu/Faculty-Bmgt1/KM/papers.nfs, Septiembre de 2003.