UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA INDUSTRIAL Y DE SISTEMAS

TESIS

"ALGORITMO VORAZ ITERATIVO CON MECANISMO DE DESTRUCCIÓN MIXTO SEMI ALEATORIO PARA LA PROGRAMACIÓN A GRAN ESCALA DE MÁQUINAS HETEROGÉNEAS EN PARALELO"

PARA OBTENER EL GRADO DE DOCTOR EN INGENIERÍA INDUSTRIAL

ELABORADO POR:

JUAN CARLOS SOTELO VILLENA

ASESOR:

JOSÉ CARLOS ALVAREZ MERINO

LIMA - PERU

2023

DEDICATORIA

A mi madre que siempre hace todo por mi bienestar

A la memoria de mi esposa que siempre me animó en este emprendimiento y que desde el cielo me acompaña en este logro

AGRADECIMIENTOS

A Dios por darme la inteligencia para servir

A mi esposa que siempre me animó en todos mis emprendimientos

A mis profesores que con sus lecciones contribuyeron en mi formación

A todas las personas de las empresas y de la academia que me apoyaron para tratar de mejor manera el tema del Scheduling y lograr los resultados de esta investigación

RESUMEN

La investigación trata el tema de la programación de máquinas en paralelo, conocido como Parallel Machine Scheduling (PMS). Se enfoca en el problema de la programación a gran escala, donde el número de trabajos es muy grande y mucho mayor que el número de recursos. La solución consiste en asignar y secuenciar los trabajos en los recursos, de modo tal, que se logren terminar a tiempo o con el menor atraso posible.

PMS es un problema clásico, tanto de la ingeniería industrial por su relación con la optimización de recursos, y de la matemática computacional de análisis combinatorio. Casos de pocos trabajos y recursos, han sido resueltos satisfactoriamente por modelos de optimización, los casos de mayor tamaño se abordan con métodos heurísticos y se logran soluciones aceptables. La programación a gran escala es un problema relativamente nuevo, pero cuya presencia aumenta debido a la tendencia de fabricación de lotes de producción más pequeños para lograr una oferta más variada de productos. La literatura sobre casos de gran escala es aún escaza.

La tesis incluye dos papers, uno con una revisión de métodos para PMS y otro donde se compara el desempeño de algoritmos heurísticos para casos de gran escala. Finalmente se desarrolla una modificación del algoritmo voraz iterativo y se comprueba empíricamente que su desempeño aumenta considerablemente.

DESCRIPTORES TEMÁTICOS

Programación de máquinas en paralelo, Algoritmo voraz, Algoritmo genético, Programación de producción, Métodos heurísticos

ABSTRACT

The research deals with the topic of Parallel Machine Programming (PMS). It focuses on the problem of large-scale scheduling, where the number of jobs is very large and much greater than the number of resources. The solution is to allocate and sequence the jobs on the resources, such that they can be completed on time or with the least possible delay.

PMS is a classic problem, both in industrial engineering due to its relation to resource optimization, and in computational mathematics of combinatorial analysis. Cases with few jobs and resources, have been satisfactorily solved by optimization models, larger cases are addressed with heuristic methods and acceptable solutions are achieved. Large-scale scheduling is a relatively new problem, but its presence is growing due to the trend towards smaller production batches for a more diverse product offering. The literature on large-scale cases is still scarce.

The thesis includes two papers, one with a review of methods for PMS and another where the performance of heuristic algorithms for large-scale cases is compared. Finally, a modification of the greedy iterative algorithm is developed and it is empirically verified that its performance increases considerably.

KEYWORDS

Parallel machine scheduling (PMS), Greedy algorithm, Genetic algorithm, Production scheduling, Heuristic methods.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El Scheduling puede entenderse en general como el problema de asignar recursos a lo largo del tiempo para realizar un conjunto de tareas de un proceso de fabricación o de servicio (Blazewicz et al., 2007). Pinedo (2016), añade que el propósito de asignar tareas a recursos en un espacio limitado de tiempo, es optimizar uno o más objetivos y que, el Scheduling debe entenderse como un proceso de toma de decisión que ocurre de manera regular en muchas empresas de manufactura y de servicios. Baker (2009) considera que las empresas siempre tienen un conjunto de tareas por realizar y un conjunto disponible de recursos para realizar las tareas y que en general el problema de Scheduling es determinar en qué momento, se realizará cada tarea y qué recurso la realizará, todo ello considerando una serie de restricciones.

Debido a la gran diversidad de actividades empresariales, las tareas y los recursos de una empresa pueden adoptar muy diversas formas. Los recursos en una fábrica pueden ser las máquinas o las personas, en un aeropuerto serían las pistas de aterrizaje, en una construcción serían las cuadrillas de peones, en un computador serían los procesadores, y así por el estilo (Pinedo, 2016). Como los primeros trabajos de Scheduling se dieron en el ambiente de la fabricación los recursos suelen denominarse máquinas y las tareas trabajos (Baker,2009).

En función de las diferentes configuraciones que pueden tener las máquinas se pueden establecer diferentes categorías de problemas de Scheduling (Pinedo, 2016). Estas categorías son descritas en el capítulo II como parte del marco referencial de la tesis. La investigación se enfoca en la categoría PMS o programación de máquinas en paralelo.

CAPÍTULO I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1. LA PROBLEMÁTICA DEL SCHEDULING

La PMS aborda una clase de problema de combinatoria compleja del tipo NP-Hard, que consiste en asignar un conjunto de trabajos a un conjunto de máquinas dentro de un cierto periodo de tiempo (Baker, 2009). En el terreno de la complejidad computacional, se consideran problemas NP-Hard a cierta clase de problemas donde no se ha encontrado un algoritmo de tiempo polinomial para encontrar la solución (Pinedo, 2016). Esto significa que no se ha encontrado como determinar el tiempo de solución en función del tamaño del problema. Los problemas de la categoría PMS son comunes en las industrias y representan un aspecto relevante en la eficiencia y eficacia de los procesos. Las empresas tratan de cumplir con las fechas de entrega y a la vez reducir los costos. Ambos objetivos requieren una mejor utilización de los recursos en el proceso.

Para la resolver problemas de PMS se han utilizado métodos exactos basados en ramificación y acotación y programación matemática. Sin embargo, estos enfoques han mostrado limitaciones para problemas de gran escala (Sotelo et al., 2016). Desde la segunda mitad del siglo XX se han publicado muchísimos trabajos sobre PMS. La exploración que Gantt realiza a mediados del siglo XX sobre secuenciación de trabajos sobre una y dos máquinas constituye uno de los trabajos pioneros sobre PMS (Gacias et al., 2011).

La investigación en PMS ha pasado de aplicar métodos exactos de optimización, para resolver casos pequeños con un número reducido de trabajos y máquinas, a explorar casos de mayor escala mediante la aplicación de métodos heurísticos que generan soluciones aproximadas al óptimo (Liao et al., 2012). Las investigaciones

más recientes evidencian un marcado interés de los investigadores aplicando diversos métodos heurísticos como el algoritmo voraz iterativo (Ying y Cheng, 2010; Lin, 2011; El Idrissi, 2018), la búsqueda tabú (Bozorgirad y Logendran, 2012; Saricicek y Celik, 2011), el recosido simulado (Saricicek y Celik, 2011; Laha, 2012), el algoritmo genético (Balin, 2011; Li et al., 2011) o algoritmo inmune inspirado (Diana et al., 2015). También se encuentran trabajos con técnicas combinadas como el propuesto por Lin et al (2016) que combina el algoritmo voraz iterativo con una lista tabú.

Como consecuencia de la globalización de la economía el mercado privilegia la diferenciación, lo que genera lotes de producción más pequeños y de múltiples variantes de producto. Esta situación presiona a los fabricantes a resolver con mayor frecuencia, secuencias de producción con miles de trabajos sobre decenas o centenas de máquinas. En este escenario es evidente la necesidad de lograr soluciones aceptables en tiempos razonables (Sotelo et al., 2016).

En el contexto descrito, la investigación estudia el problema de PMS a gran escala, donde generalmente el número de trabajos a programar es muchísimo mayor al número de máquinas, considerando una serie de restricciones como plazos de entrega, eficiencia o rendimiento de las máquinas, tiempos de acondicionamiento, etc. Si bien el problema de PMS ha sido ampliamente estudiado (Sotelo, 2013), el caso de PMS a gran escala no tiene aún muchos trabajos.

1.2. UN CASO DE PMS A GRAN ESCALA

El PMS es un problema complejo de optimización combinatoria (Lin et al., 2016), típico en diversos ambientes de producción. Para ilustrar el caso de PMS a gran escala, presentamos el caso de las empresas peruanas de la industria textil-confecciones, principalmente exportadoras de prendas de vestir. Se caracterizan por vender capacidad de producción, que implica al principio, recibir el diseño de la prenda del cliente, desarrollarlo y cotizar la manufactura. Cuando, tanto el desarrollo como la cotización satisfacen las expectativas del cliente, entonces reciben el Pedido para manufacturar un cierto número de unidades de la prenda en cuestión.

El ciclo de negocio de la fabricación de prendas de vestir es muy intenso como consecuencia de la velocidad de cambio de la moda y de la tendencia de los clientes a una mayor diferenciación en el vestir, lo cual ha derivado en lotes

(órdenes) más pequeños de producción por estilo y color. Esta situación genera, que este tipo empresas reciba un alto número de pedidos a producir y tenga usualmente un backlog de entre 4000 y 8000 órdenes de costura en proceso y por producir, que tienen que programarse eficientemente para poder cumplir con las fechas comprometidas de entrega al cliente y/o prever los posibles atrasos oportunamente, de forma tal que se puedan prever acciones para reducir el atraso o negociar anticipadamente cambios de fecha de entrega.

La capacidad de producción esta representadas por líneas de producción que pueden manufacturar los diferentes productos, pero con diferentes eficiencias. En este contexto, eficiencia se entiende como la relación entre el tiempo estándar para manufacturar un cierto número de prendas y el tiempo real de manufactura. Un ejemplo sencillo que ilustra esta situación se presenta en la Tabla 1. En ella se ve claramente que algunas líneas de producción son más eficientes que otras para manufacturar ciertos tipos de producto y viceversa. También se indica que algunos tipos de producto solo pueden ser manufacturados por ciertas líneas.

Tabla 1

Ejemplo de matriz de eficiencias Línea-Producto

Línea de producción	Tipo de Producto					
	Camisa	Bata	Vestido	Polo	Casaca	Pantalón
A	80%	72%		90%	60%	
B	70%			80%	75%	
C		80%	70%	75%		68%
D		85%	62%	70%		75%

En función de la explicación anterior, el problema básico consiste en asignar cada orden de costura (trabajo) a una línea de producción (máquina) en cierta secuencia, de modo tal que se minimice el tiempo total para lograr resolver todos los trabajos dentro del plazo establecido o con el menor atraso posible.

El problema extendido incorpora las siguientes consideraciones:

- Cada trabajo tiene un plazo diferente y tiempo de inicio restringido

- Existen tiempos de preparación (Set Ups) cuando un recurso pasa de un trabajo a otro y se requiere una reconfiguración
- La disponibilidad de los recursos es variable en el tiempo
- Los trabajos pueden fraccionarse para ser resueltos por más de un recurso
- El número de trabajos por asignar es muy grande (en el orden de miles de trabajos) y es mucho mayor que el número de máquinas disponibles (generalmente algunas decenas)

1.3. FORMULACIÓN DEL PROBLEMA

¿De qué manera programar (asignar y secuenciar) altos volúmenes de órdenes de producción con diferentes plazos de entrega, sobre un conjunto de máquinas no relacionadas en paralelo, en tiempos computacionales reducidos y logrando una calidad de programa aceptable en términos de tiempo de entrega y/o menor atraso posible?

1.4. OBJETIVOS DE LA INVESTIGACIÓN

La tesis plantea un objetivo general y 3 objetivos específicos:

1.4.1. Objetivo general

Seleccionar y mejorar el desempeño de un algoritmo heurístico para la programación a gran escala de órdenes de producción sobre máquinas no relacionadas en paralelo, con el propósito de reducir el tiempo computacional y lograr soluciones de mayor calidad en términos de tiempo total de entrega y/o menor atraso posible.

1.4.2. Objetivos específicos

- Ubicar entre los principales algoritmos heurísticos, uno con mejor desempeño frente a casos de PMS a gran escala.
- Mejorar el desempeño del algoritmo identificado, incorporando cambios en los criterios de decisión para encontrar mejores soluciones.
- Validar el desempeño del algoritmo modificado mediante casos de aplicación de gran escala tomados de la industria de la confección.

1.5. JUSTIFICACIÓN

La justificación de la investigación se presenta desde dos frentes: la importancia académica del tema y el impacto económico que significa su aplicación en la empresa.

Para evidenciar la importancia académica consideremos una simple búsqueda sobre PMS en el reconocido buscador de publicaciones científicos Scholar Google. En esta búsqueda y solo desde el año 2020 se encuentran 1880 papers, que representa el 14% de un total de 13400 artículos. Lo anterior evidencia que el tema es importante y mantiene el interés de los investigadores.

Otra razón para tratar el tema es que el problema se presenta ampliamente en la industria manufacturera y es relevante en el costo del proceso. Para evidenciar este aspecto tomemos el caso de una empresa de manufactura del sector textil exportador de prendas de vestir. Este tipo de empresas, lo que vende es capacidad de producción, por tanto, el cliente espera que su pedido sea atendido oportunamente. En este contexto la programación eficiente y eficaz de la capacidad de producción es necesaria para entregar los pedidos a tiempo o prever oportunamente posibles situaciones de atraso.

En empresas que venden capacidad de producción, los recursos de producción (máquinas, personal, energía, etc.) representan parte importante de la estructura de costos y por tanto la rentabilidad del negocio depende en buena cuenta de la adecuada programación de estos recursos.

Para evidenciar el impacto económico que implica la programación adecuada de la utilización de los recursos de producción, tomemos el caso de una empresa con una planta de costura de 1000 operarios organizados en líneas de producción (las máquinas en paralelo). La jornada normal de trabajo es de 600 minutos (8 horas más 2 extras), el nivel de eficiencia promedio se estima en 80%. Con estos datos se determinan los minutos estándar a producir por día:

$$\text{Minutos estándar diarios a producir} = 600 \times 0.80 \times 1000 = 480000$$

El valor en el mercado del minuto estándar de costura es de US$ 0.06, por tanto, si por efectos de la programación de la planta de costura, se asume solo un 5% de variación (incremento o disminución) de la eficiencia de planta, se tiene un

incremento o disminución de 24000 minutos estándar diarios, cuyo valor en el mercado es de US$ 0.06 x 24000, lo que da US$ 1400 diarios. Al anualizar, considerando 52 semanas de trabajo por 6 días, se tiene una variación de US$ 449280.

El ejemplo desarrollado muestra como la calidad de la programación de los recursos impacta directamente en la economía de la empresa. Este impacto aumenta al incrementarse el tamaño de la empresa y/o incrementarse la variación de la eficiencia.

Adicionalmente, un buen programa de producción, oportunamente actualizado, impacta positivamente en todas las funciones de este tipo de empresa. El área de Producción se beneficia por un mejor aprovechamiento de los recursos que influye en una mejor productividad y en la reducción de los costos de producción. Además, puede prever anticipadamente los requerimientos de capacidad futuros. El área Comercial se beneficia porque puede conocer si los pedidos se entregaran a tiempo o negociar los atrasos previstos oportunamente, evitando costos por penalidades o cancelación de pedidos. Además, se beneficia porque al conocer como está cubierta la capacidad de planta puede orientar convenientemente la colocación de los nuevos pedidos. El área Logística se beneficia al conocer con mayor anticipación las necesidades de producción puede negociar con los proveedores en mejores condiciones, logrando reducir los costos de aprovisionamiento. Finalmente, el área de Administración y Finanzas se beneficia porque al conocer las proyecciones de despacho y aprovisionamiento podrá determinar con mayor anticipación y precisión las variaciones del flujo de caja y tomar decisiones financieras mejor informadas.

En conclusión, solucionar el problema planteado tiene un carácter integrador en la gestión empresarial y es de gran importancia por el impacto económico que implica.

1.6. HIPÓTESIS DE LA INVESTIGACIÓN

1.6.1. Hipótesis

La modificación del criterio de destrucción aleatorio del algoritmo voraz iterativo por un criterio semialeatorio, considerando siempre que sea posible la destrucción de los alfa trabajos más atrasados, permite elevar tanto el desempeño en tiempo de

ejecución como la calidad de la solución para casos de programación a gran escala, de máquinas heterogéneas en paralelo.

1.6.2. Variables

Se proponen las siguientes variables conceptuales:

Variable independiente:

- El criterio de decisión aplicado en algoritmo voraz iterativo para ubicar mejores soluciones

Variables dependientes:

- La calidad del programa generado por el algoritmo heurístico mejorado
- El desempeño del algoritmo voraz iterativo mejorado, en términos del tiempo computacional para generar la solución

CAPÍTULO II

MARCO REFERENCIAL

2.1. REVISIÓN GENERAL DE LA LITERATURA SOBRE PMS

La investigación desarrolló como primer paso, una revisión detallada sobre cómo habían evolucionado los métodos para resolver problemas de Machine Scheduling en sus diferentes variantes. Para la revisión, se seleccionaron una serie de papers de revistas indexadas y disponibles en bases de datos científicas. Como consecuencia de dicho trabajo se desarrolló un primer paper (Sotelo, 2013), publicado en la Revista Industria, Sociedad y Sistemas, del Instituto de Investigación de la Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional de Ingeniería y que presentamos a continuación:

Parallel Machine Scheduling: Una revisión de métodos

Mg. Juan Carlos Sotelo Villena
Universidad Nacional de Ingeniería
Av. Túpac Amaru 210, Lima 25, Perú
jcs@andestec.com

Resumen

En este artículo se presenta una revisión sobre las principales investigaciones realizadas sobre la programación de máquinas en paralelo (PMS), con especial énfasis en las investigaciones realizadas desde el 2009 hasta el 2012. El objetivo es mostrar el estado del arte sobre PMS. La revisión realizada revela que PMS constituye una importante categoría dentro de los problemas de scheduling en general. Los más recientes estudios muestran que el tema aun es tratado a nivel teorético y que hay un vasto campo para investigar sobre aplicaciones reales de la industria.

Palabras Clave: Programación de máquinas en paralelo, programación, secuenciación.

Abstract

In this paper presents a review of the major research done on parallel machines scheduling (PMS), with special emphasis on research conducted from 2009 through 2012. The aim is to show the state of the art on PMS. The review reveals that PMS is a major category within the overall scheduling problems. Recent studies show that the subject is treated even theoretical level and that there is much scope for research on real industry applications.

Keywords: Parallel machine scheduling, scheduling, sequencing.

1. INTRODUCCIÓN

La programación de trabajos (scheduling) es generalmente aceptada como un problema de combinatoria compleja del tipo NP-Hard [1] [2], definido como el proceso de asignar un conjunto de tareas a los recursos dentro de un periodo de tiempo. Una programación efectiva es un factor clave en la competitividad de las empresas tanto para la reducción de costos por mejor utilización de la capacidad y menores inventarios en procesos, así como para la satisfacción del cliente por cumplimiento de las fechas de entrega y mayor calidad de los productos. Todos los aspectos mencionados son dependientes de la eficiencia con que los trabajos son programados en el sistema.

Los primeros estudios sobre secuenciación de trabajos u órdenes los realiza Henry Gantt en la primera mitad del siglo XX, posteriormente en los 50s Johnson genera uno de los primeros algoritmos para la secuenciar trabajos sobre una y dos máquinas (la regla de Johnson) [3].

Los primeros estudios sobre secuenciación de trabajos u órdenes los realiza Henry Gantt en la primera mitad del siglo XX, posteriormente en los 50s Johnson genera uno de los primeros algoritmos para la secuenciar trabajos sobre una y dos máquinas (la regla de Johnson) [3].

Durante los años 60 y 70 se aplicaron técnicas de modelado basadas en programación lineal entera, que dieron soluciones óptimas, pero limitadas a problemas de tamaño reducido y condiciones ideales [5] (pocos trabajos, pocas máquinas y relativamente pocas restricciones). Posteriormente entre los 70s y 90s se consideraron algoritmos basados en heurísticas que buscaban resolver casos de mayor envergadura con el enfoque de encontrar soluciones bastante buenas sin necesariamente ser las óptimas. En la actualidad el problema se sigue estudiando y se viene aplicando técnicas basadas en algoritmos genéticos [3] [7], búsqueda Tabú [8] [9], recosido simulado [10], entre otros métodos.

A medida que ha ido evolucionando la economía las necesidades de las empresas se han ido incrementando y ahora es mucho más frecuente encontrar casos de scheduling con miles de trabajos por secuenciar sobre decenas o centenas de máquinas, por tanto el problema de obtener soluciones en tiempo reducidos ha cobrado una importancia creciente dado que la dinámica de la economía presiona a las empresas a reajustar sus programas de producción cada vez con mayor frecuencia, debido a cambios en la demanda (pedidos urgentes, cambio de prioridades), eventualidades con los proveedores de materias primas e insumos, e interrupciones en la capacidad de producción (averías, no disponibilidad oportuna de repuestos, ausencia imprevista de operarios, etc.).

2. TAXONOMÍA DE LOS PROBLEMAS DE SCHEDULING

Los problemas de Scheduling se pueden clasificar de diversas formas según el criterio que se utilice [2]. La taxonomía que se describe está en función de la organización de las máquinas y de la estructura de operaciones de los trabajos a realizar [5]:

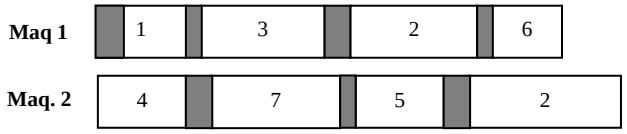
Single Machine Scheduling (SMS), es el caso más simple dado que se trata de secuenciar un conjunto de trabajos de una sola operación en una sola máquina o recurso. Las características detalladas del caso se pueden revisar en Baker [1]. Un ejemplo de este tipo de scheduling es mostrado en la figura 1.

Figura 1. Secuenciación de trabajos en una sola máquina

El número de soluciones posibles es del orden de n!, donde n es el número de trabajos. En este caso el problema es solo de secuenciación.

Parallel Machine Scheduling (PMS), La definición de Baker [1], considera en este caso un conjunto de trabajos de una sola operación que deben ser asignados y secuenciados en un conjunto de máquinas en paralelo. La figura 2 muestra un ejemplo de scheduling sobre 2 máquinas en paralelo.

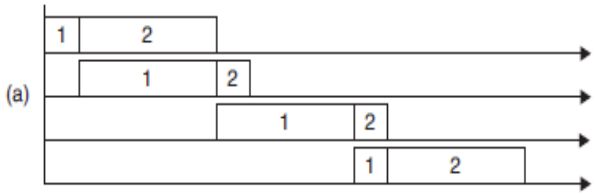
Figura 2. Ejemplo de asignación y secuenciación en dos máquinas



El número de soluciones posibles es del orden de $(n!)^m$ donde n es el número de trabajos y m es el número de máquinas. Además, el problema es de asignación y secuenciación.

Flow Shop Scheduling (FSS), según Baker [1], se considera un conjunto de trabajos que den ser procesados por un conjunto de máquinas en serie. Cada trabajo se compone de una secuencia de operaciones y cada operación de la secuencia se debe realizar en una máquina. La figura 3 muestra un ejemplo de asignación y secuenciación FSS. Una característica básica del Flow Shop Scheduling es que el flujo de trabajo es unidireccional.

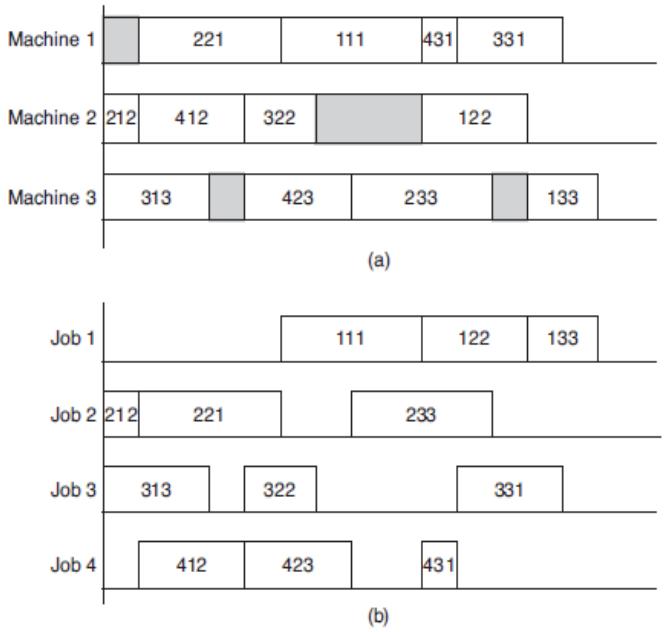
Figura 3. Ejemplo de asignación y secuenciación en Flow Shop Scheduling [1]



Job Shop Scheduling (JSS), en base a la definición de Baker [1], considera un conjunto de trabajos con una secuencia de operaciones similar al Flow Shop

scheduling que deben ser asignado y secuenciados en un conjunto de máquinas que realizan las diferentes operaciones de la secuencia. La principal distinción con respecto al Flow Shop Scheduling es que en este caso el flujo de trabajo no es unidireccional.

Figura 4. Ejemplo de asignación y secuenciación de las operaciones de 4 trabajos en Caso Job Shop [1]



Dentro de la taxonomía descrita, PMS puede aún subdividirse en dos categorías de problemas:

PMS con máquinas idénticas, constituye un caso particular pero ampliamente estudiado, dado que es muy común que por razones de capacidad una fábrica o proceso disponga de cierto número de máquinas replicadas para realizar los trabajos demandados. En este caso el problema de ubicar la máquina más apropiada es relajado y solo influye la secuencia en que se asignan los trabajos en función de las fechas de entrega y de los tiempos de set up para pasar de un trabajo a otro.

PMS con máquinas no idénticas, cuando los trabajos pueden ser resueltos por un subconjunto de las máquinas, pero con tiempo diferentes. En este caso se tiene tanto el problema de asignación, determinar que máquina es la que debe realizar el trabajo, y el problema de secuenciación, es decir como conviene ordenar los trabajos en la secuencia de cada máquina. La programación con máquinas no idénticas constituye el caso general de PMS.

Medidas de efectividad

Resolver un problema de scheduling implica, dependiendo del método utilizado, optimizar (métodos exactos) o satisfacer (métodos aproximados) una o varias medidas de efectividad. En base a la literatura revisada se pueden mencionar las siguientes medidas de efectividad:

Resolver un problema de scheduling implica, dependiendo del método utilizado, optimizar (métodos exactos) o satisfacer (métodos aproximados) una o varias medidas de efectividad. En base a la literatura revisada se pueden mencionar las siguientes medidas de efectividad:

Tiempo máximo (Makespan), es el tiempo de término del último trabajo [4]. Se busca minimizar esta medida.

Atraso (Tardiness), cada orden tiene una fecha de entrega y la idea es minimizar el atraso total, si lo hubiera [4]. En algunas ocasiones se considera un peso diferente para el atraso en cada trabajo.

Ocio de las máquinas (Idleness), se trata de minimizar el tiempo libre de las máquinas [4].

Tiempo total para completar todos los trabajos (Total Completion Time), es la suma de los tiempos requeridos para completar todos los trabajos [4]. En algunas ocasiones se agregan pesos en esta medición.

Máximo atraso (Lateness), es el atraso del trabajo que tiene el mayor atraso. Se trata de minimizar esta medida [4]. En algunas ocasiones se considera un peso diferente para el atraso en cada trabajo.

3. REVISIÓN DE LA LITERATURA EN PMS

Sobre el problema del Scheduling se vienen realizando investigaciones desde mediados del siglo XX. Debido a que la teoría del scheduling se caracteriza por un número virtualmente ilimitado de tipos de problemas [1], se han publicado miles de artículos sobre el tema. Seguidamente se referencian muy brevemente una gran cantidad de trabajos sobre machine scheduling.

En 1973, estudia el problema de minimizar el tiempo total para completar los trabajos sobre máquinas no idénticas, pero sin considerar actividad de mantenimiento. Propone que se puede obtener la solución en un tiempo polinomial si se transforma en un problema equivalente de matching de n trabajos nm posiciones de máquina. Bruno, Et al en 1974, incorporan pesos a los tiempos de proceso y determinan que el problema del PMS con setups es NP-hard para dos máquinas. Ullman en 1975, muestra que el PMS con tiempos de setup dependientes de la secuencia es un problema de tipo NP-hard. Señala además que el incremento de la complejidad por los tiempos de setup y/o las fechas de inicio diferenciadas es la razón por la que muchos estudios previos no han considerado estas restricciones.

A.H.G. Rinnooy en 1976, prueban que el problema del SMS con setups es de tipo NP-hard. Lenstra, Et al. (1977), muestran que el PMS con fechas de inicio es un problema de tipo NP-hard. R.G. Vickson en 1980, considera la minimización del makespan y del tiempo total para completar todos los trabajos, pero sobre una sola máquina. Propone un algoritmo de ramificación y acotación para la solución óptima y una heurística para una solución rápida.

Sumichrast en 1987, aplica a la PMS un algoritmo genético para minimizar el adelanto total y el atraso total en un enfoque JIT. Gupta en 1988, propone una heurística para el problema cuando los tiempos de set-up son dependientes de la secuencia de asignación. Monma, Et al. en 1989, analizan la complejidad del PMS considerando costos de setup dependientes de la secuencia de asignación. Danial y

Sarin en 1989, analizan el mismo problema y encuentran una relación entre el número de trabajos atrasados y la cantidad de recursos asignados.

Uzsoy, Et al en 1991, minimizan el máximo atraso en la PMS con restricciones de precedencia en los trabajos. Barnes, Et al en 1993, muestran que aplicar el menor tiempo ponderado de procesamiento es un simple y seguro método para generar soluciones cercanas al óptimo. J.B. Ghost en 1994, desarrollan un algoritmo de programación dinámica para resolver el problema, pero con pesos aplicados al tiempo de procesamiento. Cheng y Janiak en 1994, analizan el caso SMS considerando que los trabajos tienen fechas de inicio y con tiempos de proceso en función de los recursos consumidos. Muestran que el ordenamiento de los trabajos por fecha de inicio no decreciente da una solución óptima al problema minimizando el consumo de recursos y el makespan.

Ovacik, Et al, en 1995, propone una familia de reglas de despacho para enfrentar el problema del PMS. Además, propone unos casos de prueba que se utilizan como data sets. Nowicki y Zdrzałka, en 1995, consideran el caso de PMS con máquinas idénticas y trabajos que pueden ser adelantados y proponen un algoritmo Greedy para minimizar el costo y el tiempo total para completar los trabajos. Jacobs y Brusco en 1995, proponen un algoritmo Greedy con una simple metaheurística [11]. Cheng, Et al., en 1996, prueba que el problema de minimizar la cantidad de recursos asignados sujeto a una restricción de número de trabajos atrasados es un problema NP-Hard y propone un algoritmo de programación dinámica pseudo-polinomial.

William y Wirth en 1996, desarrollan una heurística y muestran que es efectiva en encontrar una buena solución para problemas grandes en tiempo computacional aceptable para una sola máquina. Serafini en 1996, proporciona un algoritmo para minimizar el total wighted tardiness para el caso donde cada trabajo puede ser dividido, pero sin restricción alguna y procesados independientemente en máquinas no idénticas. Wester en 1997, establece que el PMS con setups es un problema intensamente tipo NP-hard. Crauwels, Et al., en 1997, realizan múltiples pruebas de aplicación de heurísticas y encuentra que la búsqueda Tabú es la que da mejores resultados cuando se trata de un pequeño número de familias de trabajos, pero que para un número mayor de familias funcionan mejor los algoritmos genéticos.

Sule en 1997, muestra que el problema de PMS es NP-Hard y que la enumeración exhaustiva de soluciones solo es posible para casos menores a 10 órdenes y bajo ciertas condiciones. Señala que los métodos exactos son imprácticos en casos reales. Balakrishnan, Et al., en 1999, proponen un generador de datos aleatorios para obtener las fechas de entrega de los trabajos. Radhakrishnan, Et al., en el 2000, estudian la minimización del adelanto y atraso total bajo la estrategia del Justo a tiempo aplicando recosido simulado.

Dunstall, Wirth y Baker en el 2000, aplican un algoritmo de ramificación y acotación y logran resolver eficientemente problemas hasta con 70 trabajos [13]. Xing y Zhang en el 2000, proponen un algoritmo heurístico para minimizar el makespan y analizan el peor el desempeño del algoritmo en el peor caso. Además, muestran que el problema del PMS con división de trabajos cae en la categoría NP-hard. Lee y Chen en el 2000, estudian el problema de programar actividades de mantenimiento sobre máquinas idénticas en paralelo con el objetivo de obtener el mínimo tiempo total para completar los trabajos. Tratan dos versiones del problema. En un primer caso cuando el mantenimiento no puede ser realizado simultáneamente en más de una máquina y en un segundo caso donde el mantenimiento puede ser en varias máquinas

simultáneamente. Prueban que ambos problemas son NP-hard y presentan un algoritmo de ramificación y acotación que resuelve en forma exacta problemas de hasta 40 trabajos y 8 máquinas en tiempos de cómputo aceptables.

S. Webster, Et al., en el 2001, proponen un algoritmo de programación dinámica con capacidad de avance y retroceso. Muestran que el avance dinámico hacia atrás es más atractivo cuando la suma de los tiempos de proceso y setup es mayor que la suma de los pesos. Yi, Et al., en 2001, proponen una búsqueda tabú y luego una cota inferior para el PMS con setups. Liao y Liao en el 2002, proponen una búsqueda Tabú donde un mayor tiempo de setup es requerido cuando el proceso pasa de una familia de trabajos a otra, mientras que un menor tiempo de setup es aplicado cuando el cambio es entre las diferentes clases de trabajos de una misma familia.

Webster, Et al., en el 2003, proponen un algoritmo de ramificación y acotación con una solución óptima hasta 25 trabajos para un cierto número de máquinas. Chen, Et al., en el 2003, proponen un algoritmo de ramificación y acotación con el que resuelven el problema para 40 trabajos, 4 máquinas y 6 familias. Baykasoglu en el 2003, presenta un caso de aplicación de recocido simulado al PMS. Kim y Shin en el 2003, presentan un algoritmo de búsqueda Tabú restringido para el problema del PMS con restricciones, que supera en desempeño a las reglas de despacho de Ovacik, también a la búsqueda tabú básica y al simulado recocido. Liaw, Et al., en el 2003, desarrollan un algoritmo de ramificación y acotación para minimizar el atraso ponderado total en la programación de trabajos en grupo sobre máquinas en paralelo. Kim, Et al., en el 2004, estudian casos en los que cada trabajo puede ser dividido en un numero discreto de trabajos y cada uno procesados independientemente en máquinas diferentes. También señalan que hay un riesgo de incrementar los tiempos de septup por la partición de trabajos.

Bilge, Et al., en el 2004, presentan una búsqueda Tabú para minimizar el atraso total. Dunstall, Et al., en el 2005, presentan un algoritmo de ramificación y acotación para resolver con 25 trabajos y 8 familias. También proponen varias heurísticas para el PMS con setups y evalúan el desempeño de las heurísticas con relación a las cotas inferiores. Shabtay y Kaspi, en el 2006, consideran el caso PMS con máquinas idénticas y muestran que el problema con trabajos no adelantables es NP-Hard. Además, muestran que si se permite el adelanto de los trabajos el caso puede ser resuelto en un tiempo polinomial. Tahar, Et al., presentan un método basado en programación lineal para máquinas idénticas en paralelo con división de trabajos y setups dependientes de la secuencia. Kubzin y Strusevich en el 2006, estudian el caso en que la duración del mantenimiento se incrementa por deterioro en la medida que se desplaza el inicio del mantenimiento

Tahar, Et al., en el 2006, presentan un método basado en programación lineal para máquinas idénticas en paralelo con división de trabajos y setups dependientes de la secuencia. Logendran, Salmasi y Sriskandarajah en el 2006, proponen tres algoritmos de búsqueda basados en búsqueda Tabú para minimizar el tiempo total para programar todos los trabajos sobre dos máquinas para trabajos en grupo. Lin, Fowler y Montgomery en el 2006, clasifican los problemas de este tipo basados en varias estructuras de correlación. Wang y Xia en el 2007, analizan la minimización del costo total de completar todos los trabajos en un SMS. Logendran, McDonel y Smucker en el 2007, proponen varios algoritmos basados en búsqueda Tabú para minimizar el atraso total al programar trabajos en grupo sobre máquinas en paralelo considerando restricciones de disponibilidad de máquina y fechas de inicio en los trabajos.

Uzsoy en el 2007, estudia el problema PMS considerando que lo trabajos tienen diferentes fechas de inicio y aplican un AG para máquinas idénticas. Huo, et al., en el 2007, presenta un modelo heurístico multi objetivo que minimiza el número de trabajos atrasados y el máximo atraso. Rocha, Et al., en el 2008, estudian el problema PMS considerando fechas de entrega diferentes para cada trabajo y setups y máquinas no idénticas. Aplican un algoritmo GRASP. Beraldi, Et al., en el 2008, presentan un caso de PMS con división de trabajos en la industria textil, considerando setups. Allahverdi, Et al. en el 2008, realizan una revisión exhaustiva del Machine Scheduling y revelan que el PMS con restricciones adicionales como los tiempos de setup, fechas diferentes de inicio y/o entrega tiene muy pocos estudios. Shim, Et al., en el 2008, sugieren un algoritmo de ramificación y acotación para el PMS con máquinas idénticas para minimizar el total tardiness considerando que los trabajos se pueden dividir. Además, proponen uno problemas de prueba.

Wang, Et al., en el 2009, prueban que el problema del SMS se puede resolver en tiempo polinomial bajo ciertas condiciones. Su en el 2009, minimizan el adelanto y atraso total del PMS con fechas de entrega comunes a todos los trabajos. Aplican un modelo de programación binaria. Levin, Mosheiov y Sarig en el 2009, asumen que todas las máquinas deben ser mantenidas simultáneamente. También prueban que el problema es fuertemente NP-Hard si se además del número de trabajos se considera el número de máquinas. Proponen una heurística eficiente que genera muy rápidamente soluciones muy cercanas al óptimo. Hatami, Et al., en 2010, presenta un algoritmo de búsqueda tabú para el PMS con tiempos de setups dependientes de la secuencia. Lee, Et al., en el 2010, revisa el trabajo de Ovacik y encuentra que genera en algunas ocasiones soluciones no factibles. Propone también un recocido simulado restringido que obtiene un buen desempeño.

Salmasi, Logendran y Skandari, en el 2010, desarrollan un modelo de programación matemática para minimizar el total flow time para la programación de trabajos en grupo. Fanjul-Peyro y Ruiz en el 2010, proponen un algoritmo Greedy para minimizar el makespan. Salmasi, Logendran y Skandari en el 2011, extienden su trabajo anterior para minimizar el makespan en la programación de trabajos en grupo mediante la aplicación de un algoritmo hibrido de optimización por colonia de hormigas. Li & Yuan en el 2011, tratan de minimizar una función objetivo que incluye adelanto, atraso, fechas de entrega y costos de procesamiento para la programación de grupos de trabajos sobre una sola máquina.

Métodos para resolver problemas de PMS

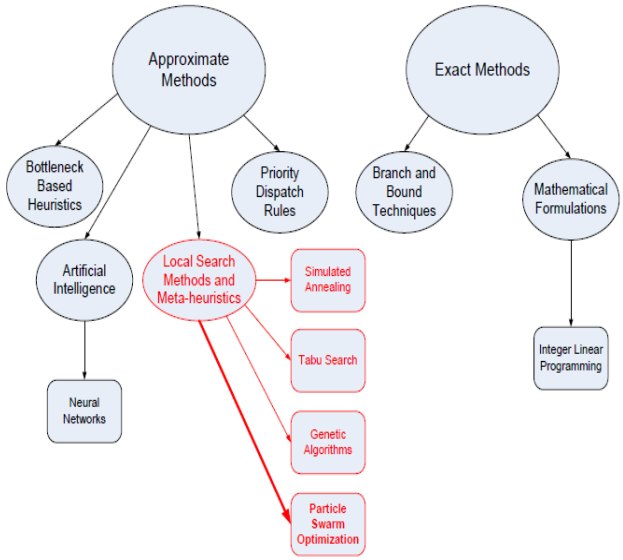
Desde la mitad del siglo pasado, con la famosa regla de Johnson para programar dos máquinas, se iniciaron las investigaciones sobre el scheduling. Los diversos métodos que se han desarrollado para resolver el problema del scheduling se pueden agrupar en dos grandes categorías, como se muestra en la Fig. 5 [2]:

Métodos exactos

Un método exacto aplica técnicas matemáticas para el modelado del problema con el propósito de minimizar una o más medidas de efectividad. En estos métodos la complejidad temporal crece exponencialmente con respecto al tamaño de la entrada [2], es decir el número de trabajos a programar. La programación lineal entera y la ramificación y acotación han sido algunos de los métodos exactos más utilizados, pero solo para problemas de muy pequeña escala.

Un problema de scheduling de n trabajos x m máquinas tiene $(n!)^m$ soluciones posibles. Esto significa que para valores de solo 20 trabajos por 10 máquinas se tienen 72615 x 10^{183} soluciones posibles, lo cual excede a la edad en segundos, atribuida al universo [2].

Figura 5. Clasificación de los métodos para resolver problemas de scheduling [7].



Este orden de magnitud hace inviable la aplicación de métodos exactos a problemas con muy alto número de trabajos y máquinas.

Métodos aproximados

El alto tiempo de cómputo y la complejidad en la formulación del problema incentivó la búsqueda de soluciones por métodos aproximados que intentaban soluciones cercanas al óptimo, pero en tiempo de cómputo razonables.

Los métodos aproximados no garantizan una solución óptima, pero son capaces de generan soluciones cercanas al optimo en tiempos de cálculo moderados [2] y, por ello son más adecuados para problemas con muy alta cantidad de órdenes.

Entre los métodos aproximados se pueden citar a los algoritmos genéticos, el recocido simulado, la búsqueda tabú y la optimización por colonia de hormigas.

4. TRABAJOS RECIENTES EN PMS

Para la revisión de los métodos del estado del arte en PMS se han seleccionado 8 artículos de journals indexados publicados desde el año 2009 hasta el año 2012. En cada caso señalamos el problema tratado las críticas del autor al estado del arte, el aporte del autor y una apreciación crítica sobre el aporte.

4.1 Aplicación de Heurística y Cota Mínima

En el 2009 Gur y Assaf publican ***A note: Simple heuristics for scheduling a maintenance activity on unrelated machines*** [4]. El estudio indica que los trabajos anteriores solo estudian el problema de PMS con máquinas idénticas. Los autores señalan, que el trabajo de Horn, si bien es con máquinas no idénticas, no considera la actividad de mantenimiento. Sobre el trabajo de Kubzin solo mencionan que es

interesante considerar que el tiempo de mantenimiento es dependiente del retraso en empezar la actividad de mantenimiento. Los autores proponen minimizar el tiempo total para completar los trabajos considerando la actividad de mantenimiento de las máquinas. Tratan dos casos del problema: Primero: considerando máquinas no idénticas y asumiendo que todas las máquinas deben ser mantenidas simultáneamente y antes de una fecha dada. Segundo: considerando máquinas no idénticas y asumiendo que el mantenimiento se puede realizar en momentos diferentes para cada máquina, pero antes de una fecha dada. Toman del trabajo de Horn, la idea de transformar el problema en uno equivalente de concordancia, pero agregan la actividad de mantenimiento.

Gur y Assaf plantean como función objetivo la minimización del tiempo total para completar todos los trabajos (Total completion Time) y para lograr el objetivo proponen introducir una heurística eficiente y una cota inferior fácil de calcular. Primero realizan un scheduling óptimo sin considerar el mantenimiento y luego acomodan el mantenimiento. Mantienen la secuencia e incorporan la actividad de mantenimiento en cada máquina, tratando que los tiempos de ocio que se generan sean los menores.

Los casos que estudian son solo de mediano tamaño (decenas o muy pocas centenas de trabajos y solo hasta tres máquinas en paralelo). Solo consideran una actividad de mantenimiento en el horizonte de programación. No consideran que los trabajos pueden estar restringidos por fecha de inicio. Es rescatable el manejo de la restricción en referencia a la necesidad de realizar actividades de mantenimiento. También es interesante su propuesta de crear los datos aleatoriamente, utilizando la técnica de Montecarlo.

4.2 Aplicación de Recocido Simulado

En el 2009 Kai Li, Ye Shi, Shan-lin Yang y Ba-yi Cheng publican ***Parallel machine scheduling problem to minimize the makespan with resource dependent processing time*** [8]. Señalan que la mayor parte de los trabajos de investigación sobre PMS asumen que los tiempos de proceso de los trabajos son valores fijos y además no consideran que los recursos son limitados. Indican que la mayoría de trabajos sobre el tema de tiempos de proceso dependientes del consumo de recursos, se ha realizado sobre la programación de una sola máquina. Proponen estudiar el problema de tiempos de proceso dependientes del consumo de recursos para el caso de máquinas idénticas en paralelo (PMS) y plantean un algoritmo basado en recosido simulado y desarrollan una serie de experimentos con data simulada para probar la bondad del algoritmo. Recosido Simulado es un proceso de búsqueda que tiene su origen en el campo de física y la ciencia de los materiales. Fue desarrollado como un modelo de simulación para describir el proceso de recosido que ocurre al condensarse la materia.

Este trabajo es uno de los pocos, que hace pruebas con casos de gran tamaño (hasta 1000 trabajos, pero solo con 8 máquinas). No consideran el caso de máquinas no idénticas. Es muy interesante como preparan la data de los experimentos aplicando simulación Montecarlo. Concluyen que el Recosido Simulado da buenos resultados sobre todo en el tiempo de cómputo y es un método recomendable para caso de gran número de trabajos.

4.3 Aplicación de Greedy Iterativo

En el 2010 Kuo-Ching Ying y Hui-Miao Cheng publican ***Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic*** [6]. El estudio considera máquinas idénticas donde los tiempos de setup son dependientes de la secuencia de asignación y los pedidos tienen fechas de llegada diferentes. Señalan que los estudios anteriores no consideran el arribo dinámico de los trabajos y tiempos de setup dependientes de la secuencia de asignación. Afirman que los algoritmos basados en búsqueda tabú son buenos, pero tienen el problema de depender de parámetros difíciles de afinar. También señalan que a pesar de que se pueden aplicar métodos exactos para resolver este tipo de problema, el enorme tiempo de computación y de consumo de memoria limita su aplicación para problemas de gran escala [6]. Proponen un algoritmo greedy iterativo para minimizar el máximo atraso. La heurística greedy iterativa es un efectivo algoritmo estocástico de búsqueda local recientemente desarrollada para problemas de optimización combinatoria. Este método fue publicado inicialmente por Jacobs y Brusco en 1995 [11].

La idea del método es crear una solución inicial completa con algún criterio y luego aleatoriamente retirar algunas asignaciones y reasignarlas de modo tal que el atraso máximo sea menor que el que existía en la solución vigente. De no ser esto posible se vuelve a la solución vigente y aleatoriamente se seleccionan otras asignaciones y se repite el proceso hasta lograr reducir suficientemente el atraso máximo o hasta que se consuma el tiempo de computo máximo asignado al proceso.

4.4 Aplicación de Recosido Simulado y Búsqueda Tabú

En el 2011 Inci y Cenk publican ***Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness*** [5]. Señalan que la PMS es un área de investigación muy popular, por su amplio rango de aplicaciones prácticas y la que la computación en paralelo ha incrementado el interés por este tema. Precisan que la PMS se puede concebir como un proceso de 2 etapas, una para determinar en qué máquina se debe realizar un trabajo y otro para determinar la secuencia dentro de cada máquina. En este trabajo se destaca que si solo fuera importante el Makespan entonces solo se tendría el problema de asignación y no el de secuenciación. Además, los autores indican que si bien se investigado mucho en PMS con máquinas idénticas, no se ha avanzado mucho en el caso de máquinas no idénticas. Finalmente señalan que hay muy pocos resultados de investigación para el caso de PMS con división de los trabajos. Inci y Cenk proponen minimizar el tiempo total de atraso considerando PMS con máquinas idénticas y división de los trabajos. Plantean su propuesta como de mucha utilidad para empresas de la industrial textil donde se procesan lotes que eventualmente se deben dividir para reducir el atraso. Considera también que existen tiempos de setup.

Proponen dos algoritmos, uno basado en Recosido Simulado y otro basado en Búsqueda tabú, en ambos casos tratan de reducir el impacto del tiempo de setup en el tiempo total ponderado. Los casos que estudian son solo de mediano tamaño (algunas decenas de trabajos). No consideran máquinas no idénticas y no consideran que la partición de los trabajos puede estar condicionada. Concluyen que el Recosido Simulado da mejores resultados que la Búsqueda Tabú y toma menos tiempo computacional.

4.5 Aplicación de Greedy y Recocido Simulado

En el 2011 Shih-Wei, Zne-Jung, Kuo-Ching y Chung-Cheng publican ***Minimization de maximum lateness on parallel machines with sequence-dependent setup times***

and job release dates [10]. Ellos señalan que la PMS con setups y trabajos restringidos por fechas de inicio y entrega es un problema que ha atraído considerable atención recientemente. También indican que a pesar de que los problemas de PMS han sido ampliamente estudiados muy poco se ha avanzado para el caso de incorporar setups dependientes de la secuencia y tiempo de inicio y entrega en los trabajos. Precisan, además, que no se ha estudiado este problema antes con métodos exactos. La investigación propone minimizar el máximo atraso (máximum lateness) considerando la PMS con máquinas idénticas, tiempos de setup dependientes de la secuencia. Implementa un algoritmo Greedy con un criterio de aceptación basado en recocido simulado. Entre las observaciones se puede mencionar que no considera máquinas no idénticas y que los casos que estudia son solo de mediano tamaño (decenas o muy pocas centenas de trabajos).

4.6 Aplicación de Algoritmo Genético

En el 2011 Savas Balin publica ***Non-identical parallel machine scheduling using genetic algorihm*** [9]. Señala que el scheduling es una función clave de la gestión de producción y que implica una decisión con un nivel de dificultad en función del número de cálculos que se tienen que realizar para determinar cómo aprovechar mejor los recursos. Indica que es muy difícil encontrar soluciones óptimas para problemas de gran escala aplicando los métodos de solución tradicionales (programación matemática). Además, señala que las reglas de despacho son solo útiles para problemas de pequeña escala. Plantea que para problemas de gran escala deben aplicarse heurísticas y entre ellas propone utilizar GA por su mejor adaptabilidad y fácil implementación. Considera el caso de minimizar el makespan para máquinas no idénticas y propone un GA con innovación en el operador de crossover.

Los algoritmos genéticos están inspirados en el proceso de evolución Darwiniano. La idea fue introducida por Holland (1975) y básicamente consiste en generar una solución inicial al problema y luego realizar iteraciones que imitan el proceso de mejora de una población por intercambio y mutación. Davies (1987) fue el primero en introducir esta técnica a problemas de scheduling [12].

Prueba su GA en una serie de experimentos computacionales con resultados auspiciosos, comparando el GA con la aplicación de la regla de despacho de mayor tiempo de proceso. Es uno de los pocos autores que trabajan casos de gran escala (hasta 500 órdenes, pero con muy pocas máquinas). No considera tiempos de setup ni fechas restringidas de inicio. Tampoco considera actividades de mantenimiento o periodos de disponibilidad restringida de los recursos. Concluyen que los resultados de los experimentos son auspiciosos y que los GA tienen la ventaja de ser muy versátiles ya que pueden ser adaptados a diferentes problemas de scheduling.

4.7 Aplicación de Búsqueda Tabú

En el año 2012, Mir Abbas Bozorgirad y Rasaratnam Logendran publican ***Sequence-dependent group scheduling problem on unrelated-parallel machines*** [7]. Los autores revisan una serie de trabajos anteriores y concluyen que el problema de PMS es muy tratado, pero con máquinas idénticas. Indican que el problema con máquinas no idénticas y con trabajos agrupados es aún poco estudiado.

Proponen optimizar simultáneamente el tiempo para completar todos los trabajos y el atraso total en la programación de grupos de trabajo sobre máquinas no idénticas en paralelo y considerando tiempos de setup. Consideran que existen grupos de trabajos

que deben programarse juntos en una máquina. También incluyen que los trabajos tienen diferentes fechas de inicio y que la disponibilidad de las máquinas es diferente. Plantean la formulación de un modelo matemático y dado que el problema es NP-Hard, proponen un algoritmo basado en búsqueda Tabú.

Sobre la base de trabajos anteriores de Logendran que muestran que la calidad de la solución final en una búsqueda Tabú es sensible a la calidad de la solución inicial, los autores proponen 2 mecanismos para identificar una buena solución inicial. Como buscan optimizar una función bi-criterio, aplican dos reglas, la regla de considerar primero a los trabajos con menor tiempo de proceso y como segunda regla la de considerar primero los trabajos con menor fecha de entrega para encontrar una solución inicial.

Por tanto, generan dos secuencias una con cada regla, luego combinan las secuencias. Como la programación es de trabajos en grupo se aplica en dos niveles primero encuentra la secuencia de grupos y luego la secuencia de trabajos dentro de cada grupo. La búsqueda tiene dos niveles: búsqueda fuera (para la secuencia de grupos en cada máquina) y búsqueda dentro para la secuencia de trabajos dentro de cada grupo. Además, propone un afinamiento de la búsqueda basada en una matriz de frecuencias de colocación de cada grupo o trabajo en una posición de máquina.

Es rescatable la consideración sobre la importancia del afinamiento de los parámetros de una búsqueda Tabú y la conclusión de que dependen de la estructura del problema. La generación aleatoria de datos (Montecarlo) para crear los experimentos de prueba también es un aporte de utilidad. El análisis de varianza luego de 972 casos experimentales también es de utilidad. La comparación con el modelo matemático corrido en CPLEX para algunos casos es una interesante forma de ver la ventaja de la búsqueda Tabú y de evidenciar de manera empírica que el problema es NP-Hard aun para casos pequeños. Los casos que estudian son solo de mediano tamaño (decenas de trabajos en el mejor de los casos) igualmente el número de máquinas es pequeño.

4.8 Aplicación de Búsqueda Tabú y reglas de dominancia

En el año 2012, Ching-Jong, Chien-Wen y Liang-Chuan publican An improved heuristic for parallel machine weighted flowtime scheduling with family set-ups times [6]. Ellos encuentran que la mayor parte de la literatura en PMS se enfoca en minimizar el makespan. Señalan que aún sin aplicar tiempos de setup el PMS es un problema NP-hard. El estudio plantea que el enfoque de solución debería centrarse minimizar el tiempo total para terminar todos los trabajos (total completion time). Proponen minimizar el tiempo ponderado total para completar todos los trabajos para el caso de PMS con máquinas idénticas, considerando que existen tiempos de setup por cada familia de trabajos. Proponen una heurística mixta para reducir el impacto del tiempo de setup en el tiempo total ponderado, combinando la Búsqueda Tabú con un par de reglas de dominancia tomadas directamente de Dunstall y Wirth [13]. Los autores no consideran el caso de máquinas no idénticas. Los casos que estudian son solo de mediano tamaño (algunas decenas de trabajos). Tampoco consideran el caso de trabajos con fechas de entrega diferentes.

CONCLUSIÓN Y TRABAJOS FUTUROS

La revisión del estado del arte permite enunciar las siguientes conclusiones:

La PMS es un problema de investigación plenamente vigente. Computacionalmente el problema está clasificado como NP-Hard, lo que significa que no es posible establecer la solución exacta en un tiempo preestablecido.

El problema básico es estudiado por cada investigador explotando algún aspecto específico. En los casos revisados un autor investiga el problema considerando los tiempos de preparación (setups), otro considera los periodos de mantenimiento de las máquinas, otro considera el fraccionamiento de las órdenes para ser procesadas en diferentes máquinas, otro considera restricciones en las fechas de llegadas de las órdenes, etc.

Lo casos revisados solo asumen casos de tamaño pequeño a mediano en referencia al número de órdenes a programar.

De todo lo anteriormente expuesto se puede comprobar dos cosas:

Primero que las características propias de cada proceso industrial, genera espacios de investigación particulares muy diversos.

Segundo, que en las investigaciones sobre el tema aún no se ha puesto énfasis en el problema con muy alta cantidad de órdenes.

Por tanto, existe un espacio de investigación muy interesante sobre PMS a gran escala con restricciones particulares a cada industria.

REFERENCIAS

[1] Baker, Keneth, ”Principles of Sequencing and Scheduling”. John Wiley & Sons, Inc. USA, 2009.

[2] Ching-Jong Liao, Chien-Wen Chao, Liang-Chuan Chen. “An improved heuristics or parallel machine wighted flowtime scheduling with family set-ups times. Journal Computer and Mathematics with Aplications, 2012

[3] Gacias, B., Artigues, C., López, P., “Parallel machine scheduling with precedence constraints and setup times”. Computers and Operations Research 37 (12), pp. 2A11-2151, 2010

[4] Gur Mosheiov, Assaf Sarig, "A Note: Simple heuristics for scheduling a maintenance activity on unrelated machines. (2009) Journal Computer and Operations Research, 2009.

[5] Inci, Saricicek y Cenk Celik, "Two Meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness. Journal Applied Mathematical Modeling, 2011.

[6] Kuo-ching Ying, Hui-Miao Cheng, “Dynamic parallel machine scheduling with sequence-dependent times using an iterated greedy heuristic. Journal Expert Systems with Applications, 2010.

[7] Mir Abbas Bozzorgirad, Rassaratnam Logendran, “Sequence-dependent group scheduling problem on unrelated-parallel machines”. Journal Expert Systems with Applications. 2012.

[8] Kai Li, Ye Shi, Shan-lin Yang, Ba-yi Cheng., “Parallel machine scheduling problem to minimize the makespan with dependent processing times”. Journal Computers & Operations Research, 2009.

[9] Savas Balin, "Non-identical parallel machine scheduling using genetic algorithm". Journal Expert Systems with Applications, 2011.

[10] Shih-Wei Lin, Zne-Jung Lee, Kuo-Ching Ying, Chung-Cheng Lu, "Minimization del maximum lateness on parallel machines with sequence-dependent setup times and job release dates". Journal Computer and Operations Research, 2011.

[11] Jacobs, L.W., & Brusco, M.J. "A local-search heuristic for large set-covering problems". Naval Research Logistics Quarterly, 1995.

[12] Davies, L. & Coombs, S. "Genetic algorithms and communications link speed design: Theoretical considerations". Grefenstette, 1987.

[13] S. Dunstall, A. Wirth, K.R. Baker. "Lower bounds and algorithms for flowtime minimization on a single machine with set-up times". Journal of Scheduling 3, 2000.

Juan Carlos Sotelo, es doctorando en Ingeniería Industrial, Maestro en Ciencias en Ingeniería de Sistemas, diplomado en Business Process Management e Ingeniero Industrial por la Universidad Nacional de Ingeniería, Lima-Perú. Tiene más de 20 años de experiencia como consultor en Procesos y Sistemas de Información y ha desarrollado algunas aplicaciones para la programación de producción para la industria textil, aplicando heurísticas y simulación discreta. Es profesor principal de la Facultad de Ingeniería Industrial y de Sistemas de la Universidad Nacional de Ingeniería y también es profesor de la Facultad de Ingeniería de la Universidad de Piura.

2.2. MARCO CONCEPTUAL

2.2.1. Orden de producción

Una orden de producción es un cierto número de unidades de un determinado producto que debe ser manufacturada para una fecha de entrega. La orden de producción puede tener una restricción de inicio.

2.2.2. Línea de producción

Una línea de producción es un conjunto de puestos de trabajos organizados para realizar las operaciones necesarias para manufacturar una orden de producción. La línea de producción está sujeta a un calendario de disponibilidad en el tiempo. Se le considera como una máquina en PMS.

2.2.3. Eficiencia Línea-Producto

La matriz de eficiencias línea-producto indica, qué líneas pueden manufacturar determinados productos y con qué nivel de eficiencia. La variedad de la matriz

representa que algunas líneas de producción son más eficientes que otras para manufacturar ciertos productos y menos eficientes en otros productos. No todas las líneas de producción pueden manufacturar todos los productos, ni todos los productos pueden ser manufacturados por todas las líneas de producción.

2.2.4. Tiempo de set up

Los tiempos de set up representan los tiempos de preparación o configuración de una línea de producción para pasar de un determinado producto a otro. Los tiempos de set up pueden ser independientes o dependientes de la línea de producción.

2.2.5. La programación de producción

Uno de los problemas centrales de la ingeniería industrial es el de la programación de producción y es el que más trabajo ha demandado a los investigadores (Abarca, 2006). La idea es asignar y secuenciar eficientemente las órdenes de producción en las máquinas o líneas de producción. Un sistema de producción con máquinas paralelas es un entorno común en el mundo real porque con una sola máquina generalmente no se puede atender la capacidad demandada, o lograr los ingresos esperados (Liao et al., 2012).

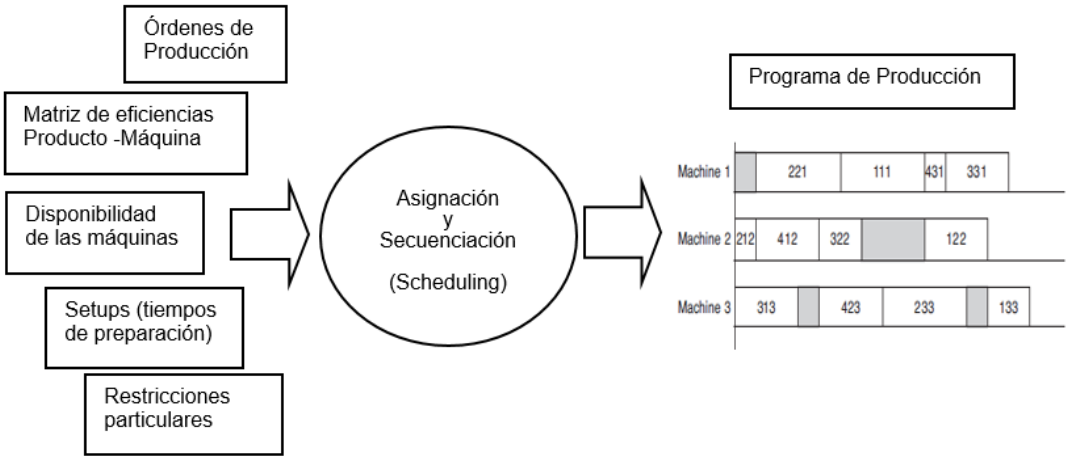
La cadena de producción para manufacturar algunos productos, como por ejemplo las prendas de vestir, puede ser bastante compleja, es decir formada por un conjunto de procesos con características diferentes, pero que deben ser adecuadamente sincronizados para reducir desperdicios de capacidad. Para esquematizar el problema de programación de producción se puede tomar un proceso cualquiera de la de la cadena bajo las siguientes consideraciones:

- Se cuenta con un conjunto de máquinas con capacidades diferentes (máquinas no relacionadas) para producir los diferentes productos, esto significa que hay máquinas más eficientes para ciertos productos y menos eficientes para otros.
- Las máquinas están disponibles un cierto número de horas por día
- Cada orden de producción requiere que la planta fabrique una cantidad de cierto producto para una fecha determinada y puede tener restricciones de inicio.
- Generalmente se requiere un tiempo de preparación (Setup) para que una máquina pueda pasar de un tipo de producto a otro.

El proceso de asignar y secuenciar las órdenes de producción sobre las máquinas, a lo largo del tiempo genera un programa de producción (Baker, 2009). La Figura 1 presenta un esquema conceptual, donde a partir de ciertas entradas y mediante un proceso de asignación y secuenciación se genera un programa de producción.

Figura 1

Conceptualización de la Programación de Producción



2.3. MARCO METODOLÓGICO

2.3.1. Tipo y diseño de investigación

La investigación es de tipo aplicada porque se enfoca en resolver un problema. El diseño de la investigación es cuasi experimental dado que utiliza muestras intencionales y se determina el efecto de la solución propuesta sobre las variables resultantes. En síntesis, se verifica el desempeño de un algoritmo de PMS de máquinas no relacionadas al manipular el criterio de búsqueda de mejores soluciones frente a casos de programación a gran escala.

2.3.2. Método de investigación

Para llevar a cabo la investigación se consideran los siguientes grandes pasos:

- Primero, revisar la literatura sobre PMS para seleccionar los algoritmos heurísticos mejor referenciados.
- Segundo, realizar un análisis comparativo del desempeño de los algoritmos seleccionados frente a casos de programación a gran escala de máquinas en paralelo, aplicando pruebas escalonadas desde 100 y hasta 5000 trabajos.

- Tercero, estudiar los criterios de decisión del algoritmo con mejor desempeño frente a casos de gran escala y proponer cambios en sus criterios de decisión, con el propósito de elevar aún más su desempeño.
- Cuarto, validar, mediante una batería de casos de prueba, si la modificación de los criterios de decisión en el algoritmo, generan un mejor desempeño.

2.3.3. Población y muestra

La población que se considera está conformada por todos los casos de programación de producción que se presentan en las empresas del sector de la industria de la confección, preferentemente exportadoras, dado que trabajan bajo pedido y se caracterizan por tener una demanda de un número muy alto de órdenes por programar.

La muestra es intencional y estará constituida por un número de casos de programación de producción, construidos a partir de la información de algunas empresas del sector de la industria de la confección exportadora.

2.3.4. Técnicas e instrumentos para la recolección y análisis de datos

En esta investigación se utilizan las siguientes técnicas e instrumentos:

- Casos de programación de producción de gran escala, tomados de empresas del sector textil-confecciones
- Técnica de Montecarlo para construir aleatoriamente casos de prueba de gran escala.
- Software de hoja de cálculo para el análisis y organización de la data de los casos de prueba.
- Lenguaje de programación de propósito general para la codificación de los algoritmos heurísticos a evaluar.
- Computadora de alto desempeño para las corridas experimentales de los algoritmos heurísticos a estudiar.

CAPÍTULO III

ANÁLISIS COMPARATIVO DEL DESEMPEÑO DE ALGORITMOS HEURÍSTICOS PARA LA PROGRAMACIÓN A GRAN ESCALA DE MÁQUINAS HETEROGÉNEAS EN PARALELO

Los resultados encontrados al revisar la literatura evidenciaron, lo que sospechábamos, que aún no hay estudios suficientes sobre PMS para casos de gran escala y por tanto existe un espacio de investigación muy fértil en este tema, que es la motivación principal de la tesis. Siguiendo el método trazado para esta tesis, se seleccionaron dos de los algoritmos más referidos en la literatura y muy ampliamente aplicados: el Algoritmo Genético propuesto por Balin (2011) y el Algoritmos Voraz Iterativo propuesto por Ying y Cheng (2010). Se desarrolló un proceso de experimentación para someter a ambos algoritmos a casos de gran escala. El experimento aumentaba progresivamente el número de trabajos a programar, desde 100 hasta 5000. La experimentación tenía como propósito descubrir como evolucionaba el desempeño de ambos algoritmos, al ser sometidos a casos de programación a gran escala. La medida para la comparación del desempeño fue el atraso máximo. Se encontró que para casos de gran escala el algoritmo voraz iterativo propuesto por Ying y Cheng (2010) logró un mejor desempeño.

Las consideraciones, resultados y conclusiones de esta parte de la investigación se presentaron en el VIII Congreso Internacional de Computación y Telecomunicaciones COMTEL 2016, de la Universidad Inca Garcilaso de la Vega, y la ponencia fue publicada en la memoria del COMTEL 2016 (Sotelo et al., 2016). A continuación, reproducimos el paper correspondiente.

Estudio comparativo de métodos heurísticos para programación de trabajos a gran escala sobre máquinas heterogéneas en paralelo

Juan Carlos Sotelo Villena, Walter Alberto Becerra Otoya, Luis Felipe Medina Aquino
jsotelo@uni.edu.pe, walter.becerra.o@uni.pe, lmedina@uni.edu.pe
Universidad Nacional de Ingeniería, Perú
Av. Túpac Amaru 210, Rímac
Lima – Perú

Resumen: *En este trabajo se compara el desempeño de dos algoritmos heurísticos para la programación de trabajos a gran escala sobre máquinas no idénticas en paralelo (Parallel Machine Scheduling - PMS). Se presenta de forma genérica la evolución, tanto del problema de scheduling como de los métodos aplicados para su solución. Se revisan una serie de trabajos recientes sobre PMS principalmente de algoritmos heurísticos. Actualmente la globalización de la economía y la tendencia de los clientes a individualizarse, han generado un escenario de lotes de producción más pequeños con múltiples variantes del producto, lo que se traduce en la necesidad cada vez mayor de casos de scheduling de gran escala, con cientos o miles de trabajos por programar. Por tanto, el problema de obtener soluciones aceptables en tiempos razonables, se torna crítico para que las empresas puedan ajustar y modificar, cada vez con mayor frecuencia, sus programas de producción. En este contexto en este trabajo se revisan e implementan a nivel experimental, el algoritmo Greedy Iterativo de Ying-Cheng y el algoritmo Genético de Savas. Se analiza como varía la calidad de la solución a medida que la carga de trabajos se incrementa desde 100 hasta 5000 trabajos. La variable aplicada para representar la calidad de la solución es el Atraso Máximo. El estudio realizado revela que el algoritmo Greedy Iterativo de Ying-Cheng se desempeña con ventaja cuando el número de trabajos tiende a ser mucho mayor. La investigación realizada evidencia que existe un amplio campo de estudio sobre temas de scheduling.*

Palabras clave: Programación de máquinas en paralelo, Programación, Secuenciación, Algoritmo genético, Algoritmo voraz.

Abstract: *In this work it is studied the performance of two heuristic algorithms for large-scale programming on non-identical parallel machines (Parallel Machine Scheduling – PMS). Generically it presents the evolution of both, the scheduling problem as the methods used to solve them. A series of recent work on PMS mainly of heuristic algorithms are reviewed. Currently, the globalization of the economy and the trend of customers to individualized, have generated a scenario of smaller production batches with multiple product variants, resulting in the growing need for scheduling cases of large scale, hundreds or thousands of jobs per schedule. Therefore, the problem of obtaining acceptable solutions at reasonable times, it becomes critical for companies to adjust and change, with increasing frequency, their production programs. In this context in this paper are reviewed and implemented on an experimental basis, the algorithm Greedy Iterative developed by Ying- Cheng and Genetic Algorithm developed by Savas. It analyzes the quality variation of the solution as the work load increases from 100 to 5000 jobs. The variable applied to represent the quality of the solution is the Maximum Delay. The study reveals that algorithm Greedy Iterative by Ying- Cheng is better when the number of jobs tends to be much higher. The research shows that there is a broad field of study on scheduling issues.*

Keywords: Parallel machine scheduling, scheduling, sequencing, genetic algorithm, greedy algorithm.

1 Introducción

La programación de trabajos sobre un conjunto de máquinas es un tipo de problema aceptado como un problema de combinatoria compleja del tipo NP-Hard [1] [2], que se define como el proceso de asignar un conjunto de trabajos a un conjunto de recursos (máquinas) dentro de un cierto periodo de tiempo. Este tipo de problema es común en las industrias y representa un aspecto importante en la eficiencia y eficacia del proceso de producción, dado que una mejor utilización de los recursos incide en una reducción de los costos y en un mejor cumplimiento de las fechas de entrega al cliente.

Los trabajos de investigación sobre PMS se inician en la primera mitad del siglo XX cuando Gantt explora la secuenciación de trabajos y desarrolla el diagrama que lleva su nombre, luego Johnson crea su famosa regla para secuenciar trabajos sobre una y dos máquinas [5].

A partir de la segunda mitad del siglo XX y con el auge de las computadoras, se desarrollan múltiples métodos o algoritmos, primero basados en métodos exactos como la programación lineal entera, pero que solo consiguen soluciones óptimas para problemas de tamaño reducido a pocas máquinas, pocos trabajos y pocas restricciones [7]. Luego se desarrollan algoritmos basados en heurísticas para casos de mayor tamaño, donde el objetivo es encontrar una solución aceptable, no necesariamente optima, en un tiempo computacional razonable. Actualmente la investigación en PMS sigue vigente y los investigadores vienen explorando variantes del problema de PMS aplicando diversos algoritmos heurísticos como búsqueda tabú [8] [11], recosido simulado [12] o algoritmos genéticos [5] [10], entre otros.

La dinámica de la economía globalizada ha generado un escenario de lotes de producción más pequeños y de múltiples variantes de producto, lo que se traduce en la necesidad cada vez mayor de tener casos de scheduling con miles de trabajos por secuenciar sobre decenas o centenas de máquinas, por tanto el problema de obtener soluciones en tiempo razonables se torna necesario para que las empresas puedan ajustar y modificar sus programas de producción cada vez con mayor frecuencia, debido a cambios en la prioridad de la demanda, eventualidades en la capacidad de la producción (averías) y eventualidades en el cumplimiento de los abastecimientos por parte de los proveedores [11].

Dada la alta complejidad que puede tomar el problema debido a la gran cantidad de datos y a las reglas muy particulares de cada sector de la industria, no es posible establecer necesariamente un mejor método o algoritmo y el estado del arte refiere a un conjunto importante de algoritmos heurísticos que compiten en la solución del problema bajo ciertas condiciones. Entre los algoritmos heurísticos más referidos tenemos a los algoritmos genéticos, la búsqueda tabú, el recocido simulado y los algoritmos greedy y grasp entre otros. En esta investigación se estudian un Algoritmo Greedy Iterativo y un Algoritmo Genético frente a casos de alta cantidad de pedidos.

El resto de éste paper está organizado de la siguiente manera. En la sección 2 se presenta el problema de programación de máquinas en paralelo. En la sección 3 se presenta una breve revisión de trabajos previos sobre PMS. La sección 4 describe el método de investigación aplicado. En las secciones 5 y 6 se describen de manera

breve los algoritmos greedy iterativo (Ying-Cheng) y genético (Savas). En la sección 7 se describen los experimentos realizados y los resultados son presentados en la sección 8. Las conclusiones y recomendaciones se presentan en la sección 9.

2 El problema de programación de máquinas en paralelo (PMS)

Existe una amplia variedad de problemas de scheduling y pueden ser clasificados según algunos criterios [2]. Uno de los criterios más utilizados en la literatura del tema, suele clasificar los problemas de scheduling está en función de la organización de las máquinas y de la estructura de operaciones de los trabajos a realizar [7].

2.1 Definición del problema PMS

Parallel Machine Scheduling (PMS), es una clase de problema de scheduling definido y ampliamente estudiado en la literatura de investigación [7]. Baker [1] define el problema de PMS, como un conjunto de trabajos de una sola operación que deben ser asignados y secuenciados en un conjunto de máquinas en paralelo. La figura 1 muestra un ejemplo de scheduling sobre 2 máquinas en paralelo.

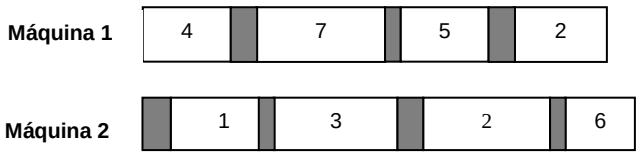


Figura 1: Ejemplo de asignación y secuenciación en dos máquinas.

En PMS número de soluciones posibles es del orden de (n!) m donde n es el número de trabajos y m es el número de máquinas, siendo un problema simultaneo de asignación y secuenciación. Dentro de la clase de problema PMS es posible distinguir además dos subclases de problema:

PMS con máquinas idénticas, que constituye un caso particular pero muy ampliamente estudiado [8], debido a que es muy frecuente que se replique máquinas para ampliar la capacidad de producción. Para este caso solo se presenta el problema de secuenciar en función de las restricciones del caso.

PMS con máquinas no idénticas, que constituye el caso general cuando los trabajos pueden ser resueltos solo por un subconjunto de las máquinas, pero con tiempo diferentes. Para este caso se tiene en simultáneo el problema de asignación y secuenciación, es necesario determinar que máquina debe realizar el trabajo y en que secuencia respecto de los otros trabajos.

2.2 Medidas de efectividad

Para evaluar la calidad de las soluciones en PMS existen varios indicadores desarrollados por los investigadores, entre ellos Tiempo Máximo (Makespan), Atraso Total, Idleness o tiempo total de ocio, Tiempo total para completar todos los trabajos y, Atraso Máximo [6]. En esta investigación se toman el atraso máximo y el tiempo total para completar todos los trabajos como los indicadores de la calidad de la solución.

3 Revisión de trabajos en PMS

Para tomar conocimiento de los algoritmos heurísticos más referidos en la literatura de investigación sobre PMS se revisaron 8 artículos de journals indexados publicados desde el año 2009 hasta el año 2012. En cada caso se indica el problema tratado, el aporte del autor y una apreciación crítica sobre el trabajo.

Gur y Assaf en el 2009 [6] estudian el problema de PMS sobre maquinas no idénticas y plantean la minimización del tiempo total para programar los trabajos incorporando una restricción para el mantenimiento de las máquinas. Estudian dos posibilidades del problema: una que asume que todas las máquinas deben ser mantenidas simultáneamente y antes de una fecha dada y otra suponiendo que el mantenimiento se puede realizar en puntos del tiempo diferentes por cada máquina, pero antes de una fecha dada.

El algoritmo de Gur y Assaf introduce una heurística eficiente y una cota inferior simple de calcular, basada en primero acomodar los pedidos loa más eficientemente posible sin considerar el mantenimiento y luego reacomodan la programación para considerar el mantenimiento tratando de desmejorar la solución inicial lo menos posible manteniendo la secuencia inicial y solo desplazando los trabajos para considerar los tiempos de mantenimiento.

Los casos que estudian son solo de mediano tamaño (decenas o muy pocas centenas de trabajos y solo hasta tres máquinas en paralelo). Solo consideran una actividad de mantenimiento en el horizonte de programación. No consideran que los trabajos pueden estar restringidos por fecha de inicio. Es rescatable el manejo de la restricción en referencia a la necesidad de realizar actividades de mantenimiento. También es interesante su propuesta de crear los datos aleatoriamente utilizando la técnica de Montecarlo.

Kai Li, Ye Shi, Shan-lin Yang y Ba-yi Cheng en el 2009 estudian el problema de PMS sobre maquinas idénticas [8] considerando tiempos de proceso dependientes del consumo de recursos y propones un algoritmo basado en Recosido Simulado y desarrollan una serie de experimentos con data simulada para probar la bondad del algoritmo. La heurística del recosido simulado es un proceso de búsqueda inspirado en el proceso que ocurre al condesarse la materia en los procesos metalúrgicos [8]. Es uno de los pocos trabajos encontrados donde el problema PMS es explorado con pruebas de tamaño considerable (hasta 1000 trabajos) pero con un número reducido de máquinas (8 máquinas), además no consideran el caso de máquinas no idénticas. En este trabajo también se destaca la preparación de data aplicando simulación Montecarlo.

Kuo-Ching Ying y Hui-Miao Cheng en el 2010 estudian el problema de PMS con máquinas idénticas [9], pero donde los tiempos de tiempos de set up son dependientes de la secuencia de asignación y los pedidos tienen fechas de llegada diferentes. Para la solución desarrollan un algoritmo Greedy Iterativo para minimizar el máximo atraso.

El algoritmo crear una solución inicial completa con algún criterio y luego inicia un proceso repetitivo de retirar aleatoriamente algunas asignaciones y reasignarlas de modo de lograr que el atraso máximo sea menor que el que existía en la solución anterior. El proceso es repetido hasta lograr un atraso máximo aceptable o hasta consumir un tiempo asignado al proceso.

Inci y Cenk en el 2011 [7] plantean que los problemas de PMS se pueden considerar como un proceso de 2 fases, una para determinar en qué máquina se debe realizar un trabajo y otro para determinar la secuencia dentro de cada máquina. Estudian el problema de minimizar el tiempo total de atraso para el caso de máquinas idénticas, pero considerando que los trabajos pueden fraccionarse y que existen tiempos de set up.

Plantean dos algoritmos, uno basado en Búsqueda Tabú y otro basado en Recosido Simulado para reducir el impacto del tiempo de set up en el tiempo total. En sus pruebas solo trabajan con casos de tamaño mediano (decenas de trabajos). Solo consideran el caso de máquinas idénticas y no tiene restricción alguna sobre la partición de trabajos. Encuentran que el Recosido Simulado les da mejores resultados que la Búsqueda Tabú.

Shih-Wei, Zne-Jung, Kuo-Ching y Chung-Cheng en el 2011 investigan el problema de PMS para minimizar el máximo atraso considerando maquinas idénticas, pero con tiempos de set up dependientes de la secuencia. Desarrollan un algoritmo Greedy que incorpora un criterio de aceptación basado en recocido simulado. Este trabajo solo considera maquinas idénticas y estudia casos que son de tamaño mediano (centenas de trabajos).

Savas Balin en el 2011 estudia el problema de PMS considerando el caso de minimizar el makespan para máquinas no idénticas y desarrolla un Algoritmo Genético, señalando que es fácil de adaptar e implementar [11]. Los algoritmos genéticos están inspirados en el proceso de evolución Darwiniano. Holland (1975) desarrollo la idea inicial y se trata de iterativamente mejorar una solución inicial por intercambio y mutación. Para problemas de scheduling, Davies (1987) fue el primero en introducir esta heurística [3].

Savas prueba su Algoritmo Genético en una serie de experimentos computacionales con resultados favorables. Es uno de los pocos autores que trabajan casos de mediano a gran tamaño (hasta 500 órdenes, pero con muy pocas máquinas), pero no considera tiempos de set up, ni fechas restringidas de inicio. Concluye que los Algoritmos Genéticos son muy versátiles y pueden ser adaptados a diferentes problemas de scheduling.

Mir Abbas Bozorgirad y Rasaratnam Logendran [10] estudian el problema PMS de optimizar simultáneamente el tiempo para completar todos los trabajos y el atraso total en la programación de grupos de trabajos sobre máquinas no idénticas, considerando tiempos de set up. Formulan de un modelo matemático y afirman que el problema es NP-Hard y desarrollan un algoritmo basado en búsqueda Tabú.

Señalan que cuando se aplica Búsqueda Tabú la calidad final depende de la calidad de la solución inicial y proponen 2 mecanismos para identificar una solución inicial adecuada. Para ello aplican dos reglas, considerar primero a los trabajos con menor tiempo de proceso y la considerar primero a los trabajos con menor fecha de entrega. Generan dos secuencias y luego las combinan. Esta lógica es aplicada en dos niveles primero secuencia de grupos y luego secuencia de trabajos dentro de cada grupo.

Es rescatable la consideración sobre la importancia del afinamiento de los parámetros de una búsqueda Tabú y la conclusión de que dependen de la estructura del problema. También es interesante considerar la generación aleatoria de datos para crear los experimentos de prueba. Estudian solo casos de tamaño mediano (decenas de trabajos) sobre muy pocas máquinas.

Ching-Jong, Chien-Wen y Liang-Chuan en el año 2012 [9]. Estudian el problema de PMS desde el enfoque de minimizar el tiempo total para completar todos los trabajos, considerando el caso de máquinas idénticas, con tiempos de set up por familias de trabajos. Desarrollan una heurística mixta combinando la Búsqueda Tabú con dos reglas de dominancia tomadas de Dunstall y Wirth [4], para reducir el impacto del tiempo de set up en el tiempo total. Los autores, solo consideran los casos de máquinas idénticas y los solo estudian casos de tamaño mediano (decenas de trabajos), no incorporan fecha de entrega diferentes.

Como consecuencia de la revisión de la literatura se encontró que los algoritmos heurísticos referidos son entre otros: Greedy Iterativo, Búsqueda Tabú, Recosido Simulado, Algoritmos Genéticos, Cota mínima y reglas de dominancia.

4 Método de investigación

El método de investigación aplicado en el presente trabajo considera 5 pasos:

Paso 1: Revisión de papers, sobre métodos y/o algoritmos para resolver problemas sobre PMS, y selección de 2 algoritmos para el estudio comparativo.

Paso 2: Revisión e implementación a nivel experimental de los algoritmos seleccionados, considerando la homogenización del problema para una comparación bajo condiciones equivalentes.

Paso 3: Prueba experimental escalonada con juegos de datos de 5000 trabajos.

Paso 3.1: Generación de data aleatoria

Paso 3.2: Aplicación incremental de la data aleatoria sobre ambos algoritmos y determinación del máximo atraso y del tiempo total asignado en cada caso.

Paso 4: Análisis de los resultados obtenidos en las pruebas experimentales.

Paso 5: Generación de conclusiones y recomendaciones.

5 Revisión e implementación del algoritmo Greedy Iterativo (Ying-Cheng)

Ying y Cheng [9], definen el problema de PMS bajo los siguientes supuestos:

- Las máquinas son idénticas
- Cada trabajo tiene una fecha de entrega (due date) y una fecha de mínimo de inicio (release date)
- Los trabajos están ordenados por fecha de mínimo inicio
- Tiempos de set up diferentes para pasar de un trabajo a otro

El algoritmo greedy iterativo propuesto por Ying y Cheng, define las siguientes variables:

N = {1,2, …, n} Conjunto de n trabajos

M = {1,2, …, m} Conjunto de m máquinas idénticas

d_i fecha de entrega del trabajo i

r_i fecha de mínimo inicio del trabajo i (ri ≤ ri + 1)

s_{ij} Tiempo de set up para pasar del trabajo i al trabajo j

s_{oi} Tiempo de set up para el primer trabajo que se coloca en una máquina

p_i Tiempo de proceso del trabajo i

C_i Fecha en que se termina el trabajo i

L_i Atraso del trabajo i (Li = Ci - di)

Define como Función objetivo: Minimizar máximo atraso (L_i).

6 Revisión e implementación del Algoritmo Genético (Savas)

Savas define el problema de PMS bajo los siguientes supuestos:

- Cada trabajo se realiza en una sola máquina.
- Las máquinas tienen diferente velocidad para realizar diferentes trabajos.
- No existen tiempos de set up para las máquinas.
- Todos los trabajos son igualmente urgentes.
- Todos los trabajos son de similar dificultad, el tiempo de procesamiento de estos depende sólo de la máquina en que se trabaja.

El algoritmo genético desarrollado por Savas considera las siguientes variables:

$V_{(i)}$: velocidad de la máquina i

$P_{(i,j)}$: Tpo de procesamiento del trabajo j cuando se realiza en la máquina i

$X_{(i,j)}$: matriz booleana (Si o No) que determina si el trabajo j se ha de realizar en la máquina i

Un trabajo se realiza solo en una sola máquina, de manera que se puede corroborar lo siguiente en la matriz booleana:

$$\sum_{i=1}^{m} X_{(i,j)} = 1, \;\; j = 1,2,3 \ldots n$$

Por condición del problema, se considera que los trabajos son de similar dificultad, por lo que, para un trabajo j se cumple:

$$P_{(i1,j)}V1 = P_{(i2,j)}V2$$

Además, para calcular los tiempos de los trabajos programados en las máquinas, necesitamos multiplicar la matriz P con la matriz Booleana X. Se cumple que: duración total de la programación:

$$C_{max} = \max_{i=1}^{m}\{\sum_{k=0}^{n} X_{(i,j)}P_{(i,j)}\}$$

Función objetivo:

$$min_{k=1}^{N}\{C_{max}(k)\},\ k = 1,\ \ldots,\ N$$

donde N es el tamaño de la población.

7 Experimentación

El objetivo fue analizar como varía la calidad de la solución en cada uno de los dos algoritmos a medida que la carga de trabajos se incrementa desde 100 hasta 5000 trabajos. Como se indicó al definir el problema de PMS, para evaluar la calidad de la solución se tomaron los indicadores de atraso máximo y tiempo total para completar todos los trabajos.

7.1 Homogenización de la definición del problema para la comparación de los algoritmos

Para el caso del algoritmo Greedy Iterativo se realizaron las siguientes relajaciones:

- No se consideran tiempos de set up. Además, se pasará de un vector de tiempos de trabajo que depende únicamente del pedido, hacia una matriz de tiempos de trabajo que depende de la máquina y el pedido.
- Todos los trabajos tienen igual release date (fecha de liberación) e igual due date (fecha de término).
- Tras una serie de pruebas previas, se vio por conveniente fijar un alfa =5, que equivale a que en cada iteración para mejorar la solución base se retiren y reacomoden 5 trabajos.
- Para el caso del Algoritmo Genético de Savas se hicieron los siguientes ajustes:
- La población de soluciones en cada iteración es igual al 10% del número de trabajos que se quiera programar.
- La fracción de cromosomas que se elegirán en el proceso de selección natural es igual al 10% del total.
- Los valores de los parámetros, para el cálculo de los valores Fitness se asignan: alfa = 0.01 y beta = 0.01, de acuerdo con recomendación del autor.

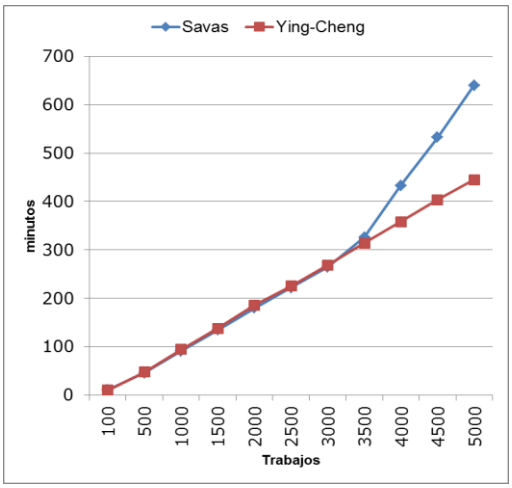
7.2 Datos para las pruebas

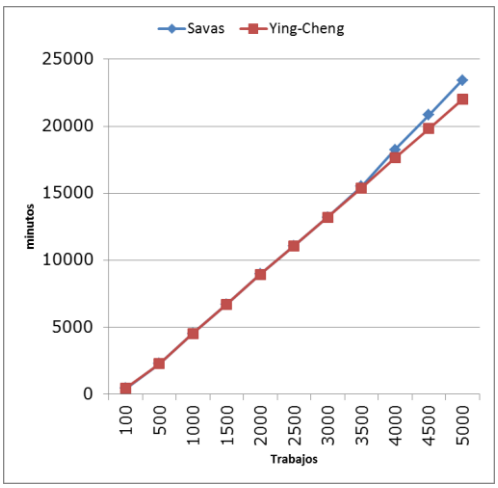
Siguiendo la tendencia encontrada en los trabajos revisados de utilizar data aleatoria para los experimentos, se generaron 3 juegos de datos aleatorios de 5000 pedidos, considerando lo siguiente:

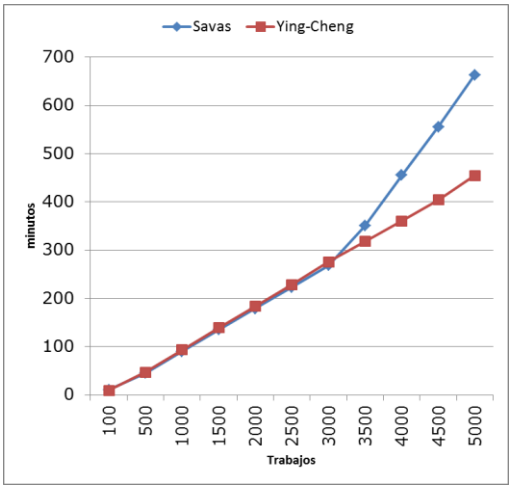
- Cantidad de máquinas consideradas: 50
- Cantidad de trabajos considerados: Incremental de 100 a 5000
- Velocidad de máquinas (aleatorio): de 1 a 3 unidades por minuto
- Tamaño de pedidos (aleatorio): de 10 a 1000 unidades.
- El tiempo para cada corrida fue de 30 minutos

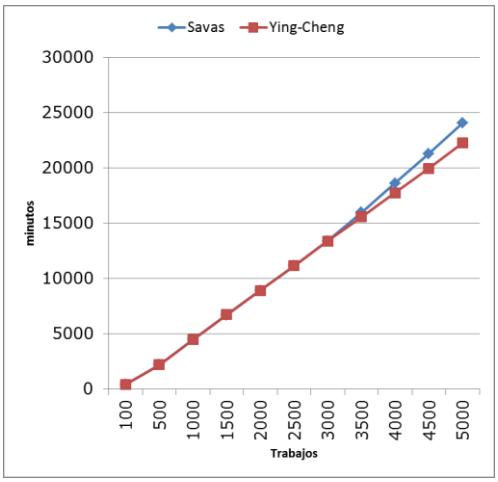
Tabla 1: Evolución del Atraso Máximo y Tiempo Total.

No.	Nro. de Trabajos	Savas		Ying Chen	
		Atraso Max (min)	Tiempo Total	Atraso Max (min)	Tiempo Total
1	100	11.11	441.07	10.10	446.39
2	500	46.00	2269.4	47.61	2266.71
3	1000	91.22	4527.08	94.58	4530.65
4	1500	134.61	6707.09	137.98	6703.17
5	2000	179.74	8965.72	185.89	8948.1
6	2500	222.09	11082.93	225.07	11083.11
7	3000	264.47	13201.59	268.96	13197.49
8	3500	325.92	15495.83	314.19	15384.28
9	4000	433.08	18233.31	358.44	17643.88
10	4500	532.69	20841.62	403.82	19810.84
11	5000	640.13	23424.15	444.69	21997.28
12	100	10.65	435.86	9.68	446.26
13	500	44.94	2206.82	46.82	2211.36
14	1000	89.86	4466.76	93.66	4471.8
15	1500	134.89	6721.05	138.93	6741.13
16	2000	178.90	8924.91	183.53	8924.86
17	2500	223.46	11149.59	228.61	11157.27
18	3000	268.21	13390.95	275.93	13411.38
19	3500	350.44	15975.11	318.03	15584.68
20	4000	455.18	18644.71	360.44	17752.82
21	4500	555.37	21320.52	404.01	19987.14
22	5000	663.29	24062.07	454.42	22262.67
23	100	11.17	470.5	10.31	470.03
24	500	48.26	2371.53	50.96	2371.45
25	1000	95.58	4751.39	102.16	4745.42
26	1500	141.62	7061.53	145.75	7047.62
27	2000	190.06	9478.75	193.29	9469.63
28	2500	236.93	11822.05	241.67	11830.91
29	3000	285.66	14262.33	289.64	14254.94
30	3500	366.95	16920.11	340.06	16634.49
31	4000	487.72	20024.82	391.84	19104.35
32	4500	572.72	22648.03	439.74	21410.03
33	5000	686.91	25579.17	484.32	23775.57









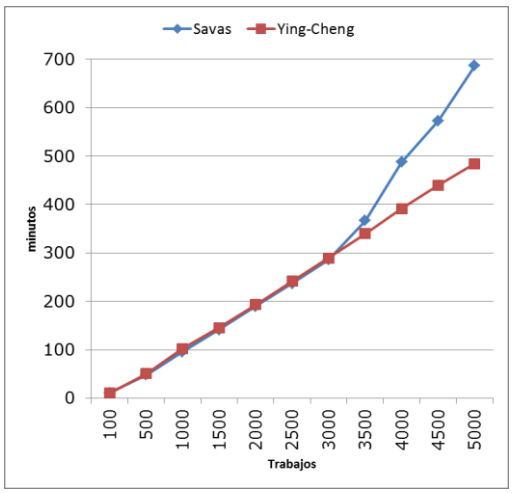


Figura 2: Evolución del Atraso Máximo al variar el número de trabajos.

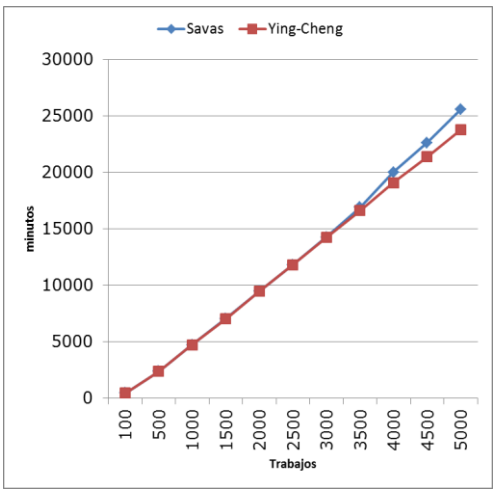


Figura 3: Evolución del Tiempo Total para s completar todos los trabajos al variar el número de trabajos.

8 Resultados

Los resultados de las pruebas experimentales realizadas con los 3 conjuntos de datos, incrementado progresivamente el número de trabajos a programar, se presentan en la tabla 1. Los resultados muestran que en el caso del algoritmo de Ying-Cheng el atraso máximo crece linealmente a diferencia del algoritmo de Savas donde el crecimiento no es lineal y a partir de los 3000 trabajos se incrementa más rápidamente que el de Ying-Cheng.

La figura 2 muestra de forma gráfica, como varía el Atraso Máximo en cada algoritmo a medida que se incrementa el número de trabajos. La figura 3 muestra que en cuanto al indicador de tiempo total para completar todos los trabajos si bien hay una diferencia favorable al algoritmo Greedy Iterativo, esta es muy pequeña.

9 Conclusiones y trabajos futuros

El trabajo realizado permite enunciar las siguientes conclusiones:

Los problemas sobre PMS son ampliamente estudiados en la literatura de investigación y cada investigador centra su trabajo en algún aspecto particular del problema, lo cual crea mayores posibilidades de investigación en el tema.

Los trabajos revisados evidencian que aún son escasos los estudios de PMS a gran escala.

Se estudiaron e implementaron a nivel experimental, dos algoritmos para resolver problemas sobre PMS: Algoritmo Greedy Iterativo (Ying-Cheng) y Algoritmo Genético (Savas).

Se analizó y se comparó como varia la calidad de la solución, expresada por el indicador de atraso máximo, de los dos algoritmos cuando se incrementa la carga de trabajo desde 100 hasta 5000 pedidos.

Para que la comparación sea bajo las mismas condiciones se relajó la definición del problema PMS de Ying-Cheng para hacerla equivalente al PMS definido por Savas.

Se encontró, en todas las pruebas que, frente a una alta carga de trabajos, el algoritmo Greedy Iterativo de Ying-Cheng es superior al algoritmo Genético de Savas.

En todas las pruebas hasta aproximadamente 3000 trabajos, no hay una diferencia notoria entre ambos algoritmos respecto del atraso máximo, pero a partir de este punto la diferencia comienza a crecer favoreciendo al algoritmo Greedy Iterativo.

Se recomienda explorar otros algoritmos heurísticos que definan el problema de PMS de manera similar a Ying-Cheng y realizar nuevos estudios de comparación.

Es recomendable también explorar la posibilidad de incluir una mejora en el algoritmo de Ying-Cheng para reemplazar la selección aleatoria de trabajos para el reacomodo por una selección de los trabajos con mayor atraso.

Referencias bibliográficas

[1] Baker, Keneth, ”Principles of Sequencing and Scheduling”. John Wiley & Sons, Inc. USA, 2009.

[2] Ching-Jong Liao, Chien-Wen Chao, Liang-Chuan Chen. “An improved

heuristics or parallel machine wighted flowtime scheduling with family set-ups times". Journal Computer and Mathematics with Aplications, 2012

[3] Davies, L. & Coombs, S. “Genetic algorithms and communications link speed design: Theoretical considerations”. Grefenstette, 1987.

[4] Dunstall, S. Wirth, A. Baker, K.R. “Lower bounds and algorithms for flowtime minimization on a single machine with set-up times”. Journal of Scheduling 3, 2000.

[5] Gacias, B., Artigues, C., López, P., “Parallel machine scheduling with precedence constraints and setup times”. Computers and Operations Research 37 (12), pp. 2A11-2151, 2010

[6] Gur Mosheiov, Assaf Sarig, "A Note: Simple heuristics for scheduling a maintenance activity on unrelated machines". (2009) Journal Computer and Operations Research, 2009.

[7] Inci, Saricicek y Cenk Celik, "Two Meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness". Journal Applied Mathematical Modeling, 2011.

[8] Kai Li, Ye Shi, Shan-lin Yang, Ba-yi Cheng. “Parallel machine scheduling problem to minimize the makespan with dependent processing times”. Journal Computers & Operations Research, 2009.

[9] Kuo-ching Ying, Hui-Miao Cheng, “Dynamic parallel machine scheduling with sequence-dependent times using an iterated greedy heuristic". Journal Expert Systems with Applications, 2010.

[10] Mir Abbas Bozzorgirad, Rassaratnam Logendran, “Sequence-dependent group scheduling problem on unrelated-parallel machines”. Journal Expert Systems with Applications. 2012.

[11] Savas Balin, “Non-identical parallel machine scheduling using genetic algorithm”. Journal Expert Systems with Applications, 2011.

[12] Shih-Wei Lin, Zne-Jung Lee, Kuo-Ching Ying, Chung-Cheng Lu, “Minimization del maximum lateness on parallel machines with sequence-dependent setup times and job release dates”. Journal Computer and Operations Research, 2011.

CAPÍTULO IV

ALGORITMO VORAZ ITERATIVO MEJORADO CON CRITERIO DE DESTRUCCIÓN SEMIALEATORIO PARA PROGRAMACIÓN A GRAN ESCALA DE MÁQUINAS HETEROGÉNEAS EN PARALELO

4.1. INTRODUCCIÓN

El análisis comparativo de los algoritmos voraz iterativo y genético, frente a casos de gran escala, en términos del atraso máximo, determinó empíricamente que el voraz iterativo propuesto por Ying y Cheng (2010) daba un mejor desempeño. Siguiendo el método de investigación de la tesis, se planteó cuestionar el mecanismo de destrucción del algoritmo voraz iterativo, debido a que es el responsable de lograr una evolución favorable en la calidad de las soluciones y además en el tiempo de ejecución. La premisa de la tesis en este punto fue, si el mecanismo de destrucción eligiera mejor que trabajos reacomodar se lograría más rápidamente una mejor solución.

Se desarrollaron 2 diseños de cambio en el mecanismo de destrucción y se realizaron 5 experimentos para determinar que diseño lograba mejores resultados. La hipótesis de trabajo en esta parte de la investigación fue que, para mejorar el desempeño de una cierta solución necesariamente habría que reacomodar los trabajos más atrasados. Se implementó un primer diseño de cambio, donde en cada evolución se reacomodaban los alfa trabajos más atrasados, pero se encontró que, si bien funcionaba en la mayoría de iteraciones, eventualmente ya no era posible mejorar la solución y el algoritmo quedaba atrapado en un óptimo local. Para superar esta barrera se plantearon 2 diseños de cambio. El primero, considera que en cada evolución se decide al azar si se reacomodan los alfa trabajos más

atrasados o simplemente se reacomodan aleatoriamente alfa trabajos. El segundo diseño considera que en cada evolución se reacomodan los alfa trabajos más atrasados y se continua así hasta encontrar un caso que no mejora. En ese caso se reacomodan alfa trabajos seleccionados aleatoriamente y se continua así hasta mejorar y luego se retorna, a la política de reacomodar los más atrasados. Se encontró que ambos diseños de cambio, lograban un mejor desempeño, sin embargo, el segundo diseño tenia los mejores resultados.

4.2. EL INDICADOR DE DESEMPEÑO EN PMS

Para evaluar el desempeño de los algoritmos que resuelven problemas de PMS se pueden encontrar en la literatura varios indicadores propuestos: Tiempo Máximo (Makespan), Atraso Total (Tardiness), Idleness o tiempo total de ocio, Tiempo total para completar todos los trabajos (Completion time) y, Atraso Máximo (maximun lateness) (Pinedo,2016) (Baker, 2009). Para esta investigación se decidió tomar como indicador el atraso máximo propuesto por Ying y Cheng (2010). Además, se consideró medir el tiempo computacional requerido para alcanzar una solución, como una medida de eficiencia.

4.3. MÉTODO DE INVESTIGACIÓN

El método de investigación aplicado en el presente trabajo considera 5 pasos:

- Paso 1: Implementación a nivel experimental del algoritmo voraz iterativo de Ying y Cheng (2010).
- Paso 2: Rediseño del mecanismo de destrucción del algoritmo voraz iterativo.
- Paso 3: Implementación a nivel experimental del algoritmo de Ying y Cheng considerando las dos modificaciones propuestas.
    - Paso 3.1: Generación de 5 juegos de datos, tomados a partir de casos reales de una empresa exportadora textil.
    - Paso 3.2: Relajación de ciertos datos como el tiempo de set up para compatibilizar los algoritmos con los datos disponibles.
- Paso 4: Experimentación y análisis de la evolución de las soluciones en cada iteración de los 3 algoritmos, original y modificados 1 y 2.

4.4. REVISIÓN E IMPLEMENTACIÓN DEL ALGORITMO VORAZ ITERATIVO

Ying y Cheng (2010), definen el problema de PMS bajo los siguientes supuestos:

- Las máquinas son idénticas
- Cada trabajo tiene una fecha de entrega (due date) y una fecha de mínimo de inicio (release date)
- Los trabajos están ordenados por fecha de mínimo inicio
- Tiempos de set up diferentes para pasar de un trabajo a otro

El algoritmo greedy iterativo propuesto por Ying y Cheng, define las siguientes variables:

$N = \{1,2,\ldots, n\}$ *Conjunto de n trabajos*

$M = \{1,2,\ldots, m\}$ *Conjunto de m máquinas idénticas*

d_i *Fecha de entrega del trabajo i*

r_i *Fecha de mínimo inicio del trabajo i* $(ri \leq ri + 1)$

s_{ij} *Tiempo de set up para pasar del trabajo i al trabajo j*

s_{oi} *Tiempo de set up para el primer trabajo que se coloca en una máquina*

p_i *Tiempo de proceso del trabajo i*

C_i *Fecha en que se termina el trabajo i*

L_i *Atraso del trabajo i* $(L_i = C_i - d_i)$

Define como Función objetivo: Minimizar máximo atraso (L_i).

El esquema general del algoritmo es el siguiente:

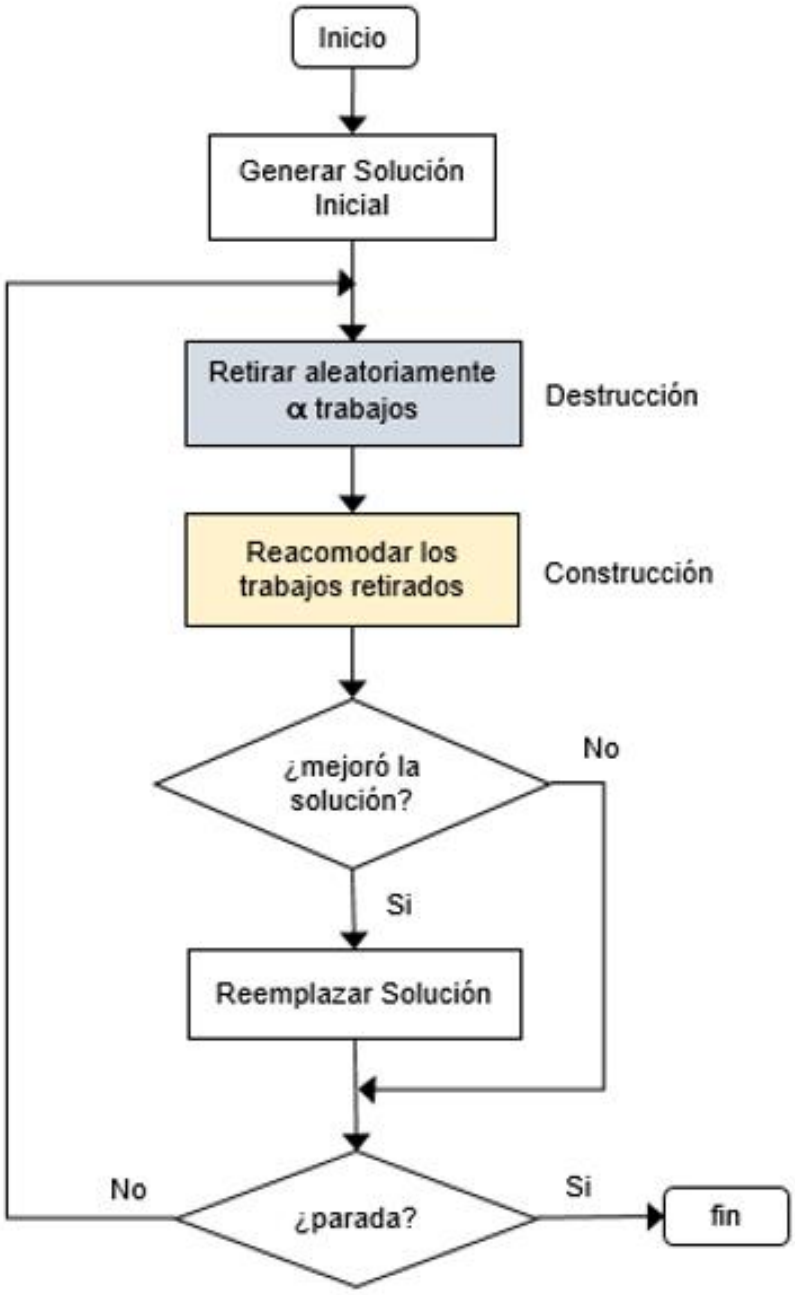
Algoritmo principal:

```
Inicio
  E0 = Generar una solución inicial
  E = E0
  Mientras no se encuentre condición de parada
    Ed = Destrucción(E)
    Ec = Construcción(Ed)
    E = Criterio de aceptación(E,Ec)
  Fin Mientras
Fin
```

La Figura 2 muestra el diagrama de flujo para mejor entendimiento de la estrategia ideada por Ying y Cheng.

Figura 2

Algoritmo voraz iterativo de Ying y Cheng



*Nota. A*daptado de Ying y Cheng (2010)

La primera etapa, donde se construye la solución Inicial, se hace de la siguiente manera:

Generación de la solución inicial:

```
Inicio
  Ordena los trabajos por fecha de entrega en ascendente
  j = 1
  Mientras (j < n)
    Asignar trabajo j a máquina con menor tiempo de
       liberación
    j = j + 1
  Fin Mientras
Fin
```

Una vez construida la solución inicial, se procede a intentar mejorarla en cada iteración mediante 3 procesos:

Destrucción de la solución:

```
Inicio
  j = 1
  Mientras (j < α)
    Seleccionar aleatoriamente un trabajo de la lista
      solución E
    Insertar el trabajo seleccionado en la lista de
      trabajos retirados ER
    Eliminar el trabajo seleccionado de la lista
      solución E
     j = j + 1
  Fin Mientras
Fin
```

Reconstrucción de la solución:

```
Inicio
  j = 1
  Mientras (j < α)
    Seleccionar el trabajo j de la lista de retirados ER
    Calcular el atraso Lj al ubicarlo en cada posición posible de
      la secuencia de asignación de cada maquina
    Insertar el trabajo j en la lista solución en la posición con
      el menor Lj
    j = j + 1
  Fin Mientras
  Eliminar Lista de retirados ER
Fin
```

Criterio de Aceptación de la nueva solución:

```
Inicio
  LE = Calcular el Máximo Li de la solución E
  LEc = Calcular el Máximo Li de la solución Ec
  Si (LEc j < LE)
    E = Ec
  Fin Si
Fin
```

El algoritmo permite establecer diversas condiciones de parada. Para este estudio se consideró un tiempo límite.

4.5. DISEÑO E IMPLEMENTACIÓN DE LA PROPUESTA DE MEJORA PARA EL ALGORITMO VORAZ ITERATIVO

En el algoritmo de Ying y Cheng (2010), discutido en el apartado anterior, se observa que la etapa de la destrucción de la solución es crítica. En esta etapa se eligen los trabajos que van a ser reubicados para buscar reducir el atraso máximo de la solución. Una elección inadecuada, de los trabajos a retirar, conllevaría a una pérdida de recursos computacionales y de tiempo, tratando de reubicarlos. Por tanto, se vio por conveniente centrar nuestra atención en mejorar el criterio de destrucción del algoritmo Voraz Iterativo. Una idea simple pero bastante eficaz que se propone, es tomar los trabajos que están más atrasados.

Si la función objetivo es:

Minimizar el máximo atraso (L_i).

Con:

L_i *Atraso del trabajo i* ($L_i = d_i - C_i$)

C_i *Fecha en que se termina el trabajo i*

d_i *Fecha de entrega del trabajo i*

El máximo L_i lo va a determinar el trabajo cuya fecha de entrega esté más distante en el pasado de la fecha de término, por lo que si lográramos reubicarlo estaríamos directamente mejorando la solución. Lo que se propone es ordenar por prioridad de retiro a los alfa trabajos más atrasados. Ahora bien, si retiráramos los alfa trabajos más atrasados en cada iteración, existe una alta probabilidad de que nos quedemos

atrapados en una solución óptima local y no en la óptima global. En otras palabras, podría suceder que la solución evolucione de manera rápida en las primeras iteraciones pero que luego se mantenga estancada en una solución que no puede mejorar retirando los alfa trabajos más atrasados. Por tanto, se propone hacer los siguientes cambios en el algoritmo voraz iterativo:

- Incorporar un variable β booleana como parámetro de la función de Destrucción. Cuando β es verdad, se retiran los α trabajos más atrasados y cuando β es falso se retiran aleatoriamente α trabajos. Se inicia el proceso inicializando a β como verdad.

- Luego de que la función Construir reacomoda los trabajos retirados, se comprueba si la nueva solución es mejor que la anterior, de ser mejor, β toma el valor de verdad y la nueva solución reemplaza a la anterior, en caso contrario β toma el valor de falso.

Algoritmo principal Mejorado:

```
Inicio
  β = verdad
  Generar Solución Inicial
  Hacer
    Destrucción de la solución(β)
    Construcción
    Si (Mejora la solución)
      Reemplazar Solución
      β = verdad
    Sino
      β = falso
    Fin Si
  Mientras (no Parada)
Fin
```

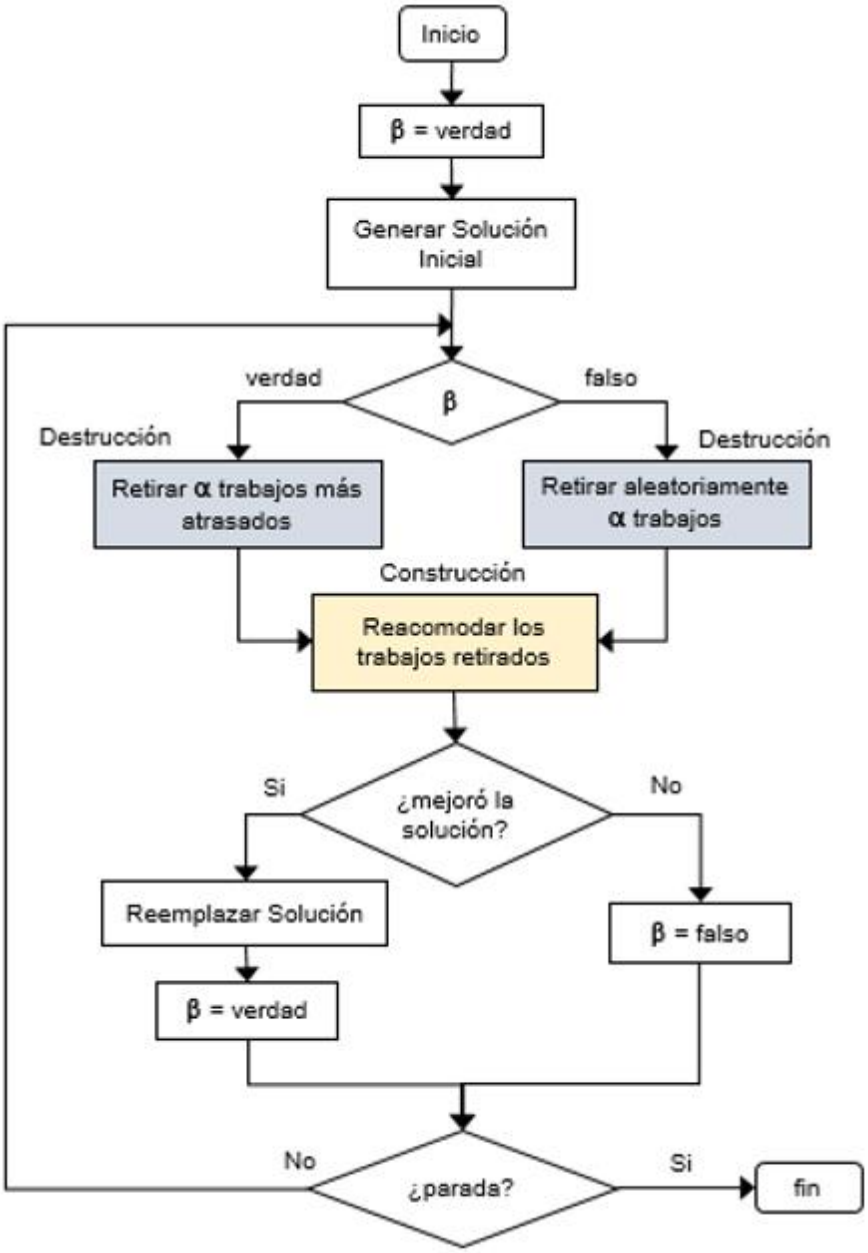
Con los cambios propuestos se logra, que siempre que retirar los α trabajos más atrasados y luego reacomodarlos, genere una mejor solución se mantenga esta política y solo se recurra al retiro aleatorio cuando no se logra una mejor solución. Este simple pero efectivo cambio, cobra mayor relevancia a medida que el número de trabajos a programar tiende a ser muy alto, dado que, al ser el espacio de búsqueda de soluciones muy alto, mejorar aleatoriamente, tiende a tener una menor

probabilidad de converger rápidamente a una mejor solución. La Figura 3 muestra un diagrama de flujo que representa el proceso descrito.

El cambio propuesto implica, además, modificar la función de Destrucción, considerando el parámetro β, para decir el retiro de los α trabajos más atrasados o el retiro aleatorio de α trabajos.

Figura 3

Algoritmo voraz iterativo mejorado por Sotelo



Función de Destrucción Mejorada:

Se incorpora un parámetro ***β*** de la función que indica qué tipo de destrucción se debe de hacer.

```
Inicio

  Si (β)
    j = 1
    L = Lista ordenada descendentemente de trabajos por atraso
    Mientras (j < α)
      Seleccionar  trabajo L[i]
      Insertar el trabajo seleccionado en la lista de trabajos
        retirados ER
      Eliminar el trabajo seleccionado de la lista solución E
      j = j + 1
    Fin Mientras
  Sino
    j = 1
    Mientras (j < α)
      Seleccionar aleatoriamente un trabajo de la lista solución E
      Insertar el trabajo seleccionado en la lista de trabajos
        retirados ER
      Eliminar el trabajo seleccionado de la lista solución E
      j = j + 1
    Fin Mientras
Fin
```

4.6. EXPERIMENTACIÓN

El objetivo fue analizar cómo mejora la calidad de la solución en la versión modificada de Ying y Cheng cuando se le somete a una carga de trabajos elevada, es decir, en el orden desde cientos a miles de trabajos. El indicador que se tomó para medir la calidad de la solución es el atraso máximo, ya que el objetivo del algoritmo es justamente minimizar este. Se hicieron pruebas con datos extraídos de casos reales de programación a gran escala.

Consideraciones para la generación de los datos de prueba

Se añadió el concepto de tipo de producto y tiempo estándar de fabricación de cada tipo de producto.

Un trabajo consta de un tipo de producto, una cantidad de este tipo de producto, una fecha de liberación y una fecha de entrega.

No se consideran tiempos de set up. Además, se pasará de un vector de tiempos de trabajo que depende únicamente del pedido, hacia una matriz de tiempos de trabajo que depende de la máquina y el tipo de producto.

Todos los trabajos tienen igual release date (fecha de liberación) aunque los trabajos tienen diferente due date (fecha de término).

Tras una serie de pruebas previas, se vio por conveniente fijar un alfa = 2% del total de trabajos, es decir, que se retiren y reacomoden el 2% del total de trabajos. No se consideraron tiempos de Setup. Es interesante notar que el valor de alfa influye directamente en el tiempo de ejecución de cada iteración.

4.7. GENERACIÓN DE LOS DATOS DE PRUEBA

Se construyó, en base a casos reales, 5 juegos de datos para las pruebas, cada uno de los cuales consideraba: una cantidad de máquinas, una cantidad de trabajos, una cantidad de tipos de producto, el primer Release Date y el último Due Date. Los datos de cada caso se muestran en la Tabla 2.

Tabla 2

Casos para las pruebas

Característica	Caso				
	1	2	3	4	5
Cantidad de Máquinas	34	45	45	45	45
Cantidad de Trabajos	321	2029	1686	1892	321
Cantidad de Tipos de producto	15	12	13	11	15
Primer Release Date	29/06/2019	24/09/2019	13/09/2019	13/11/2019	24/09/2019
Ultimo Due Date	27/08/2019	29/12/2019	30/12/2019	29/12/2019	29/12/2019

4.8. CONSIDERACIONES PARA LA COMPARACIÓN DE DESEMPEÑO

Además de la modificación propuesta en la sección 6 al algoritmo Voraz Iterativo de Ying y Cheng, se incluyó otra versión modificada con similar idea para comparar y determinar cuál proporcionaba una evolución más rápida de las soluciones.

La versión que se incluye para las pruebas se denominará Ying-Cheng Modificado 1 y la descrita en la sección número 6 será Ying-Cheng Modificado 2.

La versión modificada 2, en pocas palabras, decide qué trabajos reubicar, basado en qué tuvo éxito la iteración anterior.

Siempre que retirar los alfa trabajos más atrasados mejore la solución se volverá a usar el mismo criterio en la siguiente iteración. Si retirar los alfa trabajos más atrasados no funciona, se intentará retirar alfa trabajos de manera aleatoria. Luego de que con este criterio se mejore la solución, se vuelve a usar el criterio de retirar los alfa trabajos más atrasados.

La versión modificada 1 difiere de la versión 2 en que la decisión entre tomar los alfa trabajos más atrasados o alfa trabajos aleatorios se define de manera aleatoria y no por los resultados de las iteraciones anteriores.

Dado que estos algoritmos necesitan una condición de parada, se vio por conveniente mantener constante un tiempo prudente de ejecución: 30 minutos. Con este tiempo, incluso para el juego de dato más exigente, se tiene espacio suficiente para ver cómo evolucionan y mejoran las soluciones en cada iteración de los algoritmos considerados.

Si una curva de evolución está siempre por debajo de otra, se puede concluir que independientemente del tiempo en que se corten las ejecuciones de los algoritmos, el de la curva más baja tendrá siempre mejores soluciones disponibles.

Es importante notar que, para observar el efecto del cambio propuesto en la destrucción de la solución, no se utiliza se utiliza la búsqueda local propuesta por Ying y Cheng.

4.9. RESULTADOS

Los resultados no sólo se analizaron en cuanto a la calidad de las soluciones finales sino también a qué tan rápido se llegó a estas. Veamos primero las soluciones finales alcanzadas por cada uno de los algoritmos:

Tabla 3

Desempeño comparado de los algoritmos voraz iterativo original y modificados

Caso N°	Nro de trabajos	Ying Cheng Original		Ying Cheng Modificado 1		Ying Cheng Modificado 2	
		Tiempo	Atraso Máximo	Tiempo	Atraso Máximo	Tiempo	Atraso Máximo
1	321	21%	20.8	7%	8.4	3%	8.4
2	2029	94%	255.9	92%	163.8	10%	163.6
3	1686	100%	240.5	86%	153.0	85%	136.3
4	1892	54%	419.1	64%	331.1	64%	316.4
5	321	77%	458.3	100%	458.3	86%	326.3

En las Figuras 4 al 8, se muestra cómo evolucionan los atrasos máximos de las soluciones de cada uno de los tres algoritmos que se compararon.

Figura 4

Comparativo de evolución de soluciones Caso 1

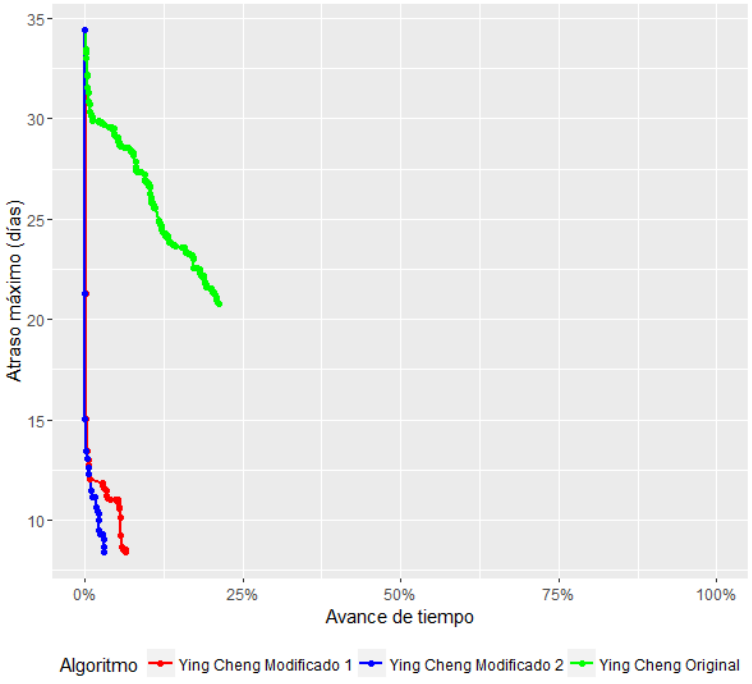


Figura 5

Comparativo de evolución de soluciones Caso 2

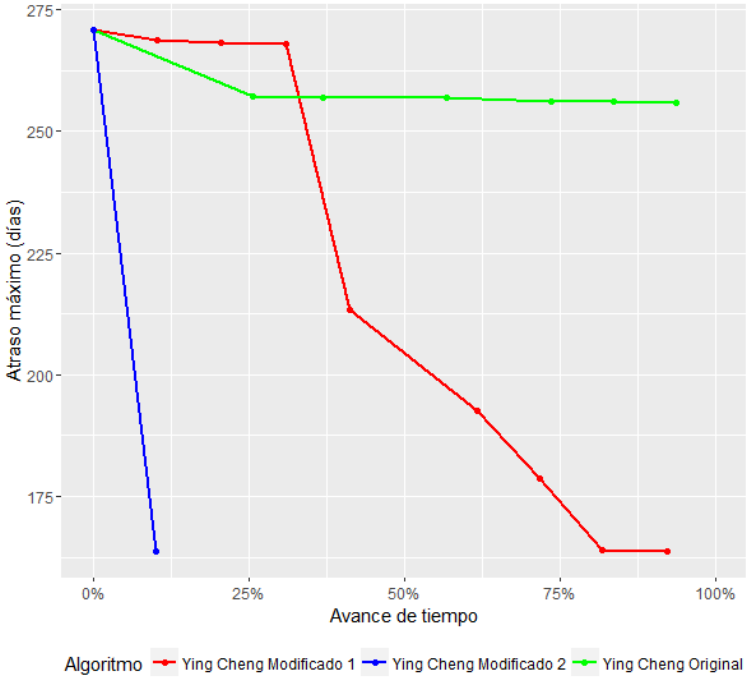


Figura 6

Comparativo de evolución de soluciones Caso 3

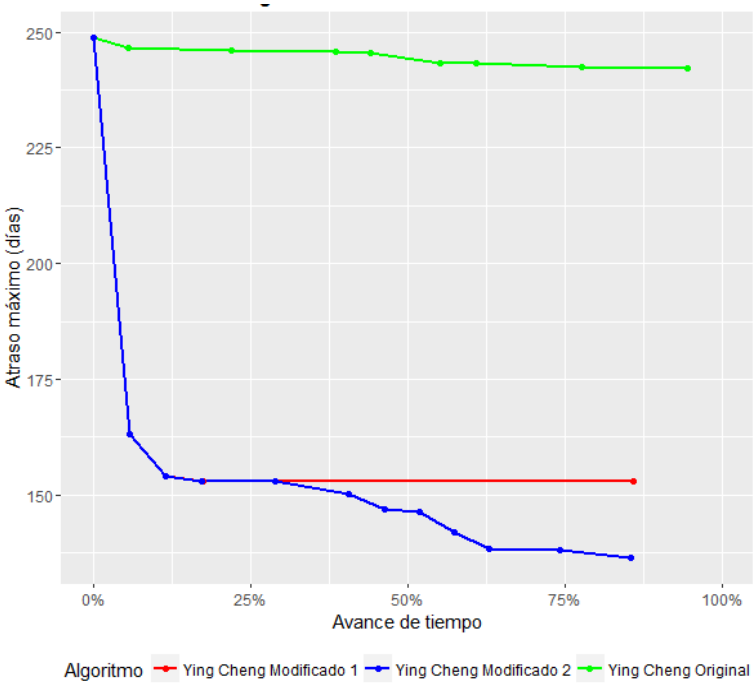


Figura 7

Comparativo de evolución de soluciones Caso 4

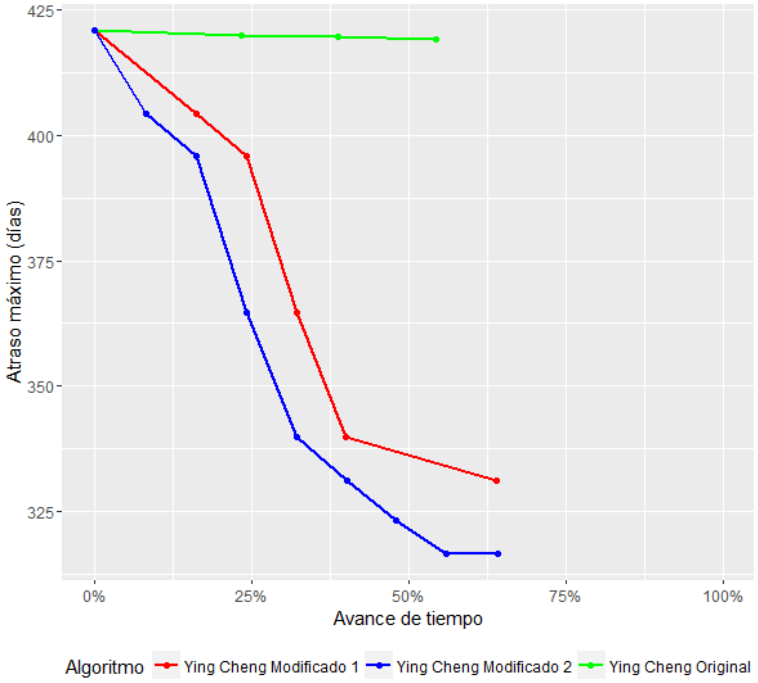
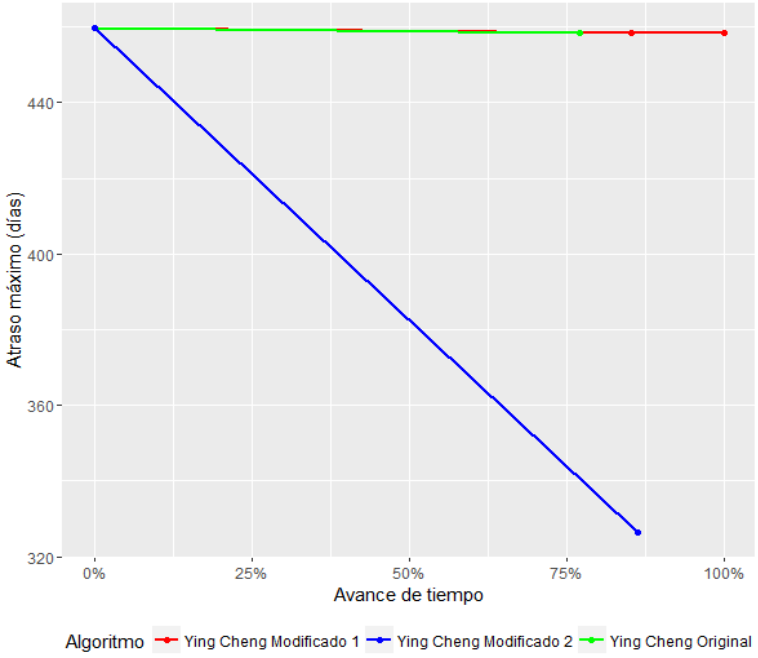


Figura 8

Comparativo de evolución de soluciones Caso 5



Como se puede observar en todos los gráficos, existe una ventaja del algoritmo de Ying-Cheng Modificado 2, frente a las otras dos versiones. No sólo la calidad de la solución final es mejor, sino que también se llega a esta de manera más rápida.

CONCLUSIONES

Sobre la revisión de la literatura:

- PMS es un problema de investigación plenamente vigente.
- Computacionalmente el problema está clasificado como NP-Hard, lo que significa que no es posible establecer la solución exacta en un tiempo preestablecido.
- El problema básico, es estudiado por cada investigador explotando algún aspecto específico. En los casos revisados un autor investiga el problema considerando los tiempos de preparación (setups), otro considera los periodos de mantenimiento de las máquinas, otro considera el fraccionamiento de las órdenes para ser procesadas en diferentes máquinas, otro considera restricciones en las fechas de llegadas de las órdenes, etc.
- Las características propias de cada proceso industrial, genera espacios de investigación particulares muy diversos.
- La mayor parte de las investigaciones tratan casos de tamaño pequeño a mediano en referencia al número de órdenes a programar.
- Aún escasa la investigación de casos de PMS a gran escala.

Sobre el análisis comparativo de desempeño de los algoritmos, voraz iterativo y genético para casos de gran escala:

- Luego de realizar 5 experimentos con casos desde 100 hasta 5000 trabajos, se encontró en todas las pruebas que, frente a una alta carga de trabajos, el algoritmo voraz iterativo de Ying-Cheng es superior al algoritmo genético de Balin.
- Las pruebas escalonadas, mostraron que hasta aproximadamente 3000 trabajos, no hay una diferencia notoria entre ambos algoritmos respecto del

atraso máximo, pero a partir de este punto la diferencia comienza a crecer favoreciendo al algoritmo voraz iterativo.

Sobre la mejora del algoritmo voraz iterativo:

- Dada la naturaleza de los problemas sobre PMS y la cantidad de variables involucradas, no es posible analizar todas de manera simultánea. En este trabajo, por ejemplo, no se ha analizado el impacto de la variación del parámetro denominado alfa, que determina la cantidad de trabajos retirados en la destrucción de la solución.
- Se analizó y se comparó como varia la calidad de la solución de los dos algoritmos, expresada por el indicador de atraso máximo. Las cinco pruebas fueron construidas en base a casos reales de programación de trabajos en una empresa textil.
- Se encontró, en todas las pruebas, frente a una alta carga de trabajos, que las modificaciones 1 y 2 al algoritmo voraz iterativo de Ying y Cheng, son superiores a su versión original.
- Además, la modificación 2 en todos los casos mantiene un resultado superior al original y al modificado 1.
- Si bien, se utilizó un tiempo de 30 minutos para las pruebas, la ventaja del algoritmo modificado 2 es tal que para cualquier instante de tiempo siempre que se haya terminado una iteración del algoritmo, la versión 2 tiene mejor solución disponible.

RECOMENDACIONES

- Realizar estudios para determinar qué proporción de los trabajos deben ser seleccionados para su reubicación (parámetro alfa). Este aspecto es importante respecto del tiempo computacional requerido. La variación del valor de alfa impacta directamente en el tiempo computacional requerido por cada iteración del algoritmo, sin embargo, no está determinado su efecto respecto de numero de iteraciones necesarias para encontrar una mejor solución.
- Se sugiere comparar el algoritmo voraz iterativo mejorado en esta investigación (modificado 2) con otros algoritmos para PMS.

REFERENCIAS

Abarca A. (2006). *El problema de la programación de órdenes de producción*. Escuela de Ingeniería Industrial y de Sistemas, Instituto Tecnológico y de Estudios Superiores de Monterrey, México.

Baker K. (2009). *Principles of Sequencing and Scheduling*. USA: John Wiley & Sons, Inc.

Balin S. (2011). *Non-identical parallel machine scheduling using genetic algorithm*. Expert Systems with Applications, Vol. 38, No. 6, pp. 6814-6821.

Blazewicz J., Ecker K., Pesch E., Schmidt G. y Weglarz J. (2007). *Handbook on Scheduling*. Springer.

Bozorgirad M.A. y Logendran R. (2012). *Sequence-dependent group scheduling problem on unrelated-parallel machines*. Expert Systems with Applications, Vol. 39, pp. 9021–9030.

Diana R.O.M., de Franca Filho M.F., de Souza S.R. y de Almeida Vitor J.F. (2015). *An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimization*. Neurocomputing, Vol. 163, pp. 94-105.

El Idrissi A., Benbrahim M., Benmansour R. y Duvivier D. (2018). *Greedy heuristics for identical parallel machine scheduling problem with single server to minimize the makespan*. MATEC Web of Conferences, Vol. 200, EDP Sciences.

Gacias B., Artigues C. y López P. (2011). *Parallel machine scheduling with precedence constrains and setup times*. Computers and Operations Research, Vol. 37, No. 12, pp. 4117-4126.

Laha D. (2012). *A simulated annealing heuristic for minimizing makespan in parallel machine scheduling*. Swarm Evolutionary and Memetic Computing, SEMCCO (Lecture Notes in computer Science), Vol.7677, Springer, Berlin.

Li K., Shi Y., Yang S.-L. y Cheng B.-Y. *Parallel machine scheduling problem to minimize the makespan with dependent processing times*. Applied Soft Computing, Vol. 11, pp. 5551-5557.

Liao C.-J., Chao C.-W. y Chen L.-C. (2012). *An improved heuristic for parallel machine weighted flowtime scheduling with family set-up times*. Computers & Mathematics with Applications, Vol. 63, No. 1, pp. 110-117.

Lin S-W., Lee Z-J., Ying K-C. y Lu C-C. (2011). *Minimization de maximum lateness on parallel machines with sequence-dependent setup times and job release dates*. Computer and Operations Research, Vol. 88, No. 33, pp. 79-90.

Lin S. W., Ying K. C., Wu W. J. y Chiang I. (2016). *Multi-objective Unrelated Parallel Machine Scheduling: A Tabu-enhanced Iterated Pareto Greedy Algorithm*. International Journal of Production Research, Vol. 54, No. 4, pp. 1110–1121.

Pinedo M.L. (2016). *Scheduling, Theory, Algorithms, and Systems*. USA: Springer.

Saricicek I. y Celik C. (2011). *Two Meta-heuristic for parallel machine scheduling with job splitting to minimize total tardiness*. Applied Mathematical Modeling, Vol. 35, No. 1, pp. 4117-4126.

Sotelo J.C. (2013). *Parallel Machine Scheduling: Una Revisión de Métodos*. Industria, Sociedad y Sistemas. Revista del Instituto de Investigación de la Facultad de Ingeniería Industrial y de Sistemas, Universidad Nacional de Ingeniería, Vol. 1, No. 1, pp. 17-26.

Sotelo J.C., Becerra W y Medina L. (2016). *Estudio comparativo de métodos para la programación de trabajos a gran escala sobre máquinas heterogéneas en paralelo*. Memoria del VIII Congreso Internacional de Computación y Telecomunicaciones, pp. 118-124.

Ying K-C. y Cheng H-M. (2010). *Dynamic parallel machine scheduling with secuence-dependent setups times using a iterated greedy heuristic*. Expert Systems with Applications, Vol. 37, No. 4, pp. 2848-2852.