ISSN 0717-3644
ISSN online 0718-221X

PROGRAMACIÓN MULTIOBJETIVO DE MÁQUINAS MOLDURERAS A TRAVÉS DE ALGORITMOS MEMÉTICOS

MULTIOBJECTIVE MOLDING MACHINE SCHEDULING USING MEMETIC ALGORITHMS

Felipe Baesler, Luis Ceballos, Milton Ramírez

RESUMEN

Este trabajo introduce un algoritmo de optimización multiobjetivo basado en la variante de la programación evolutiva denominada algoritmos meméticos (AM). Este algoritmo propuesto por los autores, combina la evolución genética con búsqueda local, al igual que los AM tradicionales, pero con la diferencia del uso de poblaciones independientes para cada objetivo. Además utiliza un mecanismo para buscar soluciones de mejor compromiso (tradeoff) en el cual se utiliza búsqueda local restringida mediante un parámetro de compromiso. Este algoritmo fue aplicado a un problema de programación de la producción en un proceso de fabricación de molduras donde es comparado con otras dos técnicas multiobjetivo disponibles en la literatura; Multiobjective Simulated Annealing (MOSA) y Multiobjective Genetic Algorithm (MOGA). El algoritmo propuesto, genera soluciones que en base a los experimentos resueltos, superan significativamente a otras técnicas utilizadas como referencia, y su validación se logra resolviendo un problema real en el cual se definen dos objetivos de interés industrial, como son: el tiempo total de fabricación (Cmax) y el atraso total. Para ambos objetivos se busca la minimización. Estos objetivos tienen impacto directo tanto en la productividad del proceso como en la capacidad de cumplimiento en las fechas de entrega de los productos a los clientes.

Palabras Clave: Multiobjetivo, Programación, Máquinas Paralelas, Algoritmos Meméticos, Máquinas Moldureras

ABSTRACT

This work presents a multiobjective optimization algorithm based on a variant of the evolutionary programming field called memetic algorithms (MA). This algorithm was proposed by the authors, combines genetic evolution with local search, in the same way as traditional MA, but with the use of independent populations for each objective, as well as a mechanism to find compromise solutions (tradeoff), where local search is performed restricted by a compromise parameter. The algorithm was applied to a scheduling problem of a molding production process and compared against two multiobjective techniques available in the literature , Multiobjective Genetic Algorithm (MOGA) and Multiobjective Simulated Annealing (MOSA). The results of the proposed approach, based on the experiments performed, outperformed the benchmark techniques based on two objectives of industrial interest, such as, the total completion time (Cmax) and the total tardiness. Both objectives are minimization. This objectives have a direct impact on the process productivity as well as the capability of delivering the goods on time.

Keywords: Multiobjective, Scheduling, Parallel Machines, Memetic Algorithms, Molding Machines

Departamento de Ingeniería Industrial, Universidad del BioBio, Concepción, Chile
Autor para correspondencia: fbaesler@ubiobio.cl
Recibido: 24 de abril de 2006.
Aceptado: 05 de septiembre 2006.

INTRODUCCIÓN

Los problemas de programación de la producción han sido estudiados intensamente en la literatura. Su carácter combinatorio hace difícil su resolución mediante métodos exactos debido a su gran complejidad computacional asociada a la categoría NP hard. La alternativa a dicho enfoque es utilizar heurísticas que intenten resolver el problema mediante la aplicación de criterios de búsqueda, obteniéndose buenos resultados en tiempo computacional aceptable. Por otro, aplicar estos métodos, en la resolución de problemas de tipo industrial, y en particular la resolución de problemas que permitan buena programación de los procesos, hace que esta área de la investigación de operaciones tenga una gran importancia en la disminución de costos de producción y en la eficiencia de los sistemas.

La remanufactura de la madera en sus diferentes procesos, presenta problemas complejos de programación de la producción. El caso que se presenta en este artículo, corresponde a la etapa de moldurado, el cual puede ser caracterizado como un problema de programación de máquinas paralelas idénticas con tiempos de setup dependientes. Los impactos que son posibles lograr mediante una buena programación son significativos, en función a los objetivos que se establecieron y en general en la productividad del sistema.

Los problemas reales por lo general son multiobjetivos por naturaleza, es decir rara vez una decisión es tomada en base a un solo criterio. Por esta razón, resulta de gran interés el desarrollo de herramientas que reflejen las necesidades industriales y que puedan entregar soluciones en un tiempo razonable. Resolver problemas reales con más de un objetivo en un tiempo razonable utilizando métodos exactos, no es posible dado el carácter combinatorio del problema. Es por esta razón que las metaheurísticas surgen como una alternativa práctica para resolver este tipo de situaciones.

El presente artículo, hace una breve descripción de metaheurísticas multiobjetivo disponibles en la literatura además de presentar una nueva herramienta propuesta por los autores, la cual se basa en la metaheurística denominada "Algoritmos Meméticos". Este algoritmo propuesto, es aplicado a un problema real de un proceso de moldurado y comparado con otras técnicas multiobjetivo. Los resultados obtenidos indican claramente las ventajas del algoritmo propuesto al ser comparado con las otras técnicas de optimización y en forma específica con la práctica utilizada por la empresa.

MARCO TEÓRICO

La optimización multiobjetivo, puede ser definida como un problema de optimización que presenta dos o más funciones objetivo. El inconveniente principal que presenta este tipo de problemas en relación a un modelo de objetivo único, radica en la subjetividad de la solución encontrada. Un problema multiobjetivo no tiene una solución óptima única, mas bien, genera un conjunto de soluciones que no pueden ser consideradas diferentes entre si. De esta manera el conjunto de soluciones óptimas es denominado Frontera de Pareto. Esta frontera de soluciones contiene todos los puntos que no son superados en todos los objetivos por otra solución. Este concepto lleva el nombre de dominancia, por esta razón la frontera de Pareto consiste solo de soluciones no dominadas. Una solución domina a otra si y sólo si, es al menos tan buena como la otra en todos sus objetivos y es mejor en al menos uno de ellos.

Para un vector de funciones objetivo que se desea minimizar,

Maderas. Ciencia y tecnología 8(3): 183-192 , 2006

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), \text{K} \quad , f_k(\vec{x}))$$

Un candidato $\vec{x}_1$ domina a $\vec{x}_2$ ($\vec{x}_1 \leq \vec{x}_2$) Si:

$$f_i(\vec{x}_1) \leq f_i(\vec{x}_2) \quad \forall i \in \{1, \ldots, k\}$$
$$Y$$
$$\exists i \in \{1, \ldots, k\} : f_i(\vec{x}_1) < f_i(\vec{x}_2)$$

La decisión final con respecto a que solución seleccionar de esta frontera de puntos dependerá de la perspectiva de cada tomador de decisiones (TD). Es decir, dependerá del nivel de compromiso que un tomador de decisiones otorga a cada objetivo dentro de su particular análisis. En los últimos años, la mayoría de los herramientas utilizadas para enfrentar este tipo de problemas se ha orientado hacia los enfoques metaheurísticos. Sin duda las técnicas mas utilizadas están ligadas al concepto de algoritmos evolutivos donde se destacan los algoritmos genéticos (AG). Los algoritmos genéticos son métodos adaptativos que pueden ser utilizados para implementar búsquedas y problemas de optimización. Ellos están basados en los procesos genéticos propios de los organismos biológicos, codificando una posible solución a un problema en un "Cromosoma" compuesto por una cadena de bits o caracteres. Estos cromosomas representan individuos que son llevados a lo largo de varias generaciones, en forma similar a las poblaciones naturales, evolucionando de acuerdo a los principios de selección natural propuestos por "Charles Darwin" en su libro "El Origen de las Especies". Emulando estos procesos, los algoritmos genéticos son capaces de "Evolucionar" soluciones a problemas del mundo real.

Los Algoritmos genéticos utilizan una analogía directa del fenómeno de evolución en la naturaleza. Trabajan con una población de individuos, cada uno representando una posible solución a un problema dado. A cada individuo se le asigna una puntuación de adaptación, dependiendo de que tan buena fue la respuesta al problema. A los más adaptados se les da oportunidad de reproducirse mediante cruzamiento con otros individuos de la población, produciendo descendientes con características de ambos padres. Los miembros menos adaptados poseen pocas probabilidades de que sean seleccionados para la reproducción, y desaparecen o mueren. Una nueva población de posibles soluciones es generada mediante la selección de los mejores individuos de la generación actual, emparejándolos entre ellos para producir un nuevo conjunto de individuos. Esta nueva generación posee una proporción más alta de características poseídas por los mejores miembros de la generación anterior. De esta forma, a lo largo de varias generaciones, las características buenas son difundidas a lo largo de la población mezclándose con otras. Favoreciendo el emparejamiento de los individuos mejor adaptados, es posible recorrer las áreas más prometedoras del espacio de búsqueda. Si el algoritmo genético ha sido diseñado correctamente, la población convergerá a una solución óptima o casi óptima del problema. Mayor información sobre teoría general de los algoritmos genéticos puede ser encontrada en (Goldberg, 1989). Algunos ejemplos de la aplicación de esta técnica a problemas de ingeniería se pueden encontrar en (Gen et. al, 1997) y (Haupt et. al, 1998).

En el ámbito multiobjetivo es posible clasificar el desarrollo de este tipo de metaheurísticas en tres grandes áreas, las cuales en forma general se pueden describir como: AGs sin elitismo, AGs con elitismo y AGs hibridizados con búsqueda local o también llamados, algoritmos meméticos (AM). En el primer grupo existe un enfoque de dominancia de soluciones como elemento de selección de cromosomas durante el proceso de búsqueda. De esta manera, los cromosomas dentro de una generación son divididos en grupos dentro de los cuales no existe dominancia. A cada grupo, se le asigna la misma función de adaptación o (fitness) debido a que no hay diferencia entre soluciones no dominadas. De esta manera se

busca una convergencia hacia la formación de una frontera de Pareto como solución al problema. Este fitness puede ser modificado para privilegiar zonas de la frontera que se encuentran menos pobladas y evitar así la concentración de soluciones en zonas particulares de la frontera de Pareto, este fenómeno se denomina formación de nichos. El primer enfoque de AG multiobjetivo bajo este esquema corresponde a MOGA (Multi-Objetive Genetic Algorithm) de Fonseca & Fleming (1993). Posteriormente han surgido muchos AG multiobjetivos basados en este enfoque en el cual se utiliza la dominancia como elemento de búsqueda. Algunos ejemplos corresponden a Nondominated Sorting Genetic Algorithm (NSGA), Srinivas & Deb (1994), y Niched-Pareto Genetic algorithm (NPGA), Horn et al. (1994).

En el segundo grupo el enfoque corresponde a técnicas que incorporan el elitismo dentro de su proceso de búsqueda. Generalmente estas técnicas mantienen una población externa en la cual se almacenan soluciones de elite, las cuales son introducidas en cada generación para mejorar la calidad de los cruzamientos entre cromosomas. Algunas técnicas dentro de esta categoría corresponden a Strength Pareto Evolutionary Algorithm (SPEA), Zitzler & Thiele (1999), NSGA II, Deb et al. (2000), MOMGA II, Zydallis et al. (2001), SPEA II, Zitzler et al. (2002) entre otros. El tercer grupo corresponde a los algoritmos meméticos, el que se expone a continuación.

ALGORITMOS MEMÉTICOS

Los algoritmos Meméticos AM, corresponden a un área de la programación evolutiva en la cual se combinan dos tipos de mecanismos de búsqueda, como son, los algoritmos genéticos y la búsqueda local. La base filosófica que sustenta está metaheurística radica en la idea de que un individuo es capaz de transformar información cultural, es decir aprendida durante su existencia, a un formato genético que le permita traspasar esta información a las generaciones siguientes por la vía biológica. Este concepto, es aprovechado para ser incorporado a la inteligencia artificial como un algoritmo genético tradicional, al cual se le incorpora algún tipo de mecanismo de búsqueda local. De esta manera, es posible hacer una analogía, en la cual el algoritmo genético representa el proceso evolutivo de los individuos y la búsqueda local se asocia a elementos culturales a los cuales cada individuo puede verse sometido durante su vida. Es así como un cromosoma antes de pasar a la generación siguiente, puede tomar una forma diferente cuando se le incorpora información aprendida durante su existencia y puede traspasar a sus descendientes la información por la vía genética. Esta herramienta ha sido utilizada con éxito en diferentes tipos de problemas preferentemente del tipo combinatorio, Jaszkiewicz (2004). En particular, aplicaciones en el ámbito multiobjetivo combinatorio es posible destacar a Ishibuchi y Kaige (2004) en problema de la mochila, Jaszkiewicz (2004) cubrimiento de conjuntos, Buriol y Franca (2004), vendedor viajero, Drezner (2003), problema de asignación cuadrática.

En el área de problemas combinatorios asociados a la programación de la producción también existen aplicaciones exitosas de algoritmos meméticos. La mayoría de los trabajos se concentran en el problema de taller de flujo (FlowShop) de los cuales se destaca Ishibuchi et al. (2003)

METODOLOGÍA

El algoritmo memético propuesto por los autores, combina la evolución genética con búsqueda local al igual que los AM tradicionales, pero con la diferencia del uso de poblaciones independientes para cada objetivo, además de un mecanismo para buscar soluciones de mejor compromiso (tradeoff) en el cual se utiliza búsqueda local restringida mediante un parámetro de compromiso o empeoramiento. Este parámetro determinado por el usuario define el empeoramiento máximo permitido para un objetivo cuando se realiza mejoramiento local en dirección de otro objetivo.

Este enfoque considera a cada objetivo en forma independiente durante todo el proceso evolutivo, sin intercambio de individuos entre poblaciones. De esta manera, es posible aplicar el algoritmo genético en formato tradicional (objetivo simple), sin necesidad de adaptarlo a un espacio multiobjetivo mediante la incorporación de pesos, dominancia y fitness compartido (shared fitness), según el enfoque a utilizar.

Sin duda esta visión del problema tiene la ventaja de conservar la estructura tradicional del algoritmo genético, logrando así una simplificación al problema. El inconveniente de enfrentar el problema de esta manera, radica en la convergencia hacia los extremos de la frontera de Pareto, dado que cada población independiente buscará las mejores soluciones para solo un objetivo. Con el fin de remediar este problema y esparcir las soluciones en una rango que cubra la mayor cobertura de la frontera de Pareto es necesario incorporar un mecanismo que genere el *trade off o compromiso* entre los objetivos. En este punto, se incorpora el proceso de búsqueda local, con el fin de lograr una mejor dispersión de las soluciones. Esto se logra seleccionando un porcentaje de las mejores soluciones encontradas en cada población de la generación actual, con el fin de aplicarles un mejoramiento local en la dirección de un objetivo diferente al que le correspondió evolucionar genéticamente. De esta manera se logra mantener la solución encontrada genéticamente, mas una solución no dominada mediante el proceso de búsqueda local. Así es posible aumentar el número de soluciones no dominadas en la generación actual. Como concepto de vecino se entiende una solución que se encuentre en el entorno cercano de la solución actual, y generalmente se traduce en modificarla levemente para generar un punto en el vecindario de ésta. En el contexto de programación de la producción se utilizó el procedimiento de inserción, es decir, se selecciona un trabajo al azar y se inserta en una posición aleatoria dentro de la secuencia que representa el cromosoma al cual se aplica la búsqueda local. Este procedimiento es tradicionalmente utilizado como vecino en problemas del ámbito combinatorio. Cuando el proceso de evolución genética culmina, se reconstruye la frontera de Pareto final con las soluciones no dominadas de todas las poblaciones incluidas en el análisis. De esta manera el tomador de decisiones está en condiciones de seleccionar la alternativa que considere mas adecuada desde su punto de vista particular. Este enfoque se denomina articulación posterior de preferencias del tomador de decisiones, dado que sus inclinaciones son consideradas en la etapa final del proceso, y no al principio como los casos de algoritmos que requieren definir ponderaciones para cada objetivo a optimizar antes de comenzar el proceso de búsqueda.

La descripción general del algoritmo se muestra a continuación. El operador de mejoramiento local se presentó en el formato de minimización a modo de ejemplificación del caso de estudio que se muestra a continuación en el cual se utilizan dos objetivos de este tipo. Para el caso de maximización basta modificar la dirección de las desigualdades del algoritmo.

1) Generar Poblaciones iniciales Pi para cada i=1, 2,....Nº de objetivos
2) Seleccionar individuos, método de la ruleta, en base a fitness fi para cada población Pi
3) Aplicar operador de mejoramiento local a individuos seleccionados en 2) en cada población Pi en base a objetivo fj , para $i \neq J$
4) Aplicar operador de cruzamiento a individuos seleccionados en cada Población Pi
5) Aplicar operador de mutación en Población Pi
6) Si el criterio de término se cumple entonces construir frontera final con soluciones no dominadas de todas las poblaciones Pi, en caso contrario ir a 2

Operador de Mejoramiento (minimización)
FOR v=1 to N° máximo de vecinos
Para individuo X generar vecino Y_v
IF $fj(X)/ fj(Y_v) \geq fi(Y_v)/ fi(X)$ **THEN**
Agregar Y_v a lista de vecinos (L)

NEXT v
Seleccionar Y_v de lista L con Máximo $fj(\mathbf{X})/ fj(\mathbf{Y}_v)$
Hacer $X=Y_v$
IF empeoramiento de $fi(X) \geq$ empeoramiento máximo **THEN** Salir
ELSE ir a 1
END

Caso de Estudio

El algoritmo descrito anteriormente fue aplicado a un problema real de la industria maderera, específicamente al proceso de fabricación de molduras. Para esto se utilizó un set de datos reales que corresponde a un problema de dos máquina moldureras iguales y un lote de 24 órdenes de trabajo. Este problema presenta tiempos de setup dependientes, es decir cuando se decide producir un producto nuevo (moldura), es necesario hacer cambios en las máquinas, los cuales generan tiempos muertos. El tiempo necesario para la realización del setup, depende del tipo de producto precedente en el programa de producción. Como objetivos para este problema se considera la minimización del tiempo total de fabricación (Cmax) y minimización del atraso total. Estos objetivos tienen impacto directo tanto en la productividad del proceso como en la capacidad de cumplimiento en las fechas de entrega de los productos a los clientes. Utilizando la nomenclatura clásica de programación de la producción, el problema antes descrito se define como, $P_2 \mid Sij \mid Cmax, \Sigma Tj$, es decir un problema de dos máquina paralelas idénticas con tiempos de setup dependientes y con objetivos de minimización del Cmax y atraso total. A continuación se explica en más detalle el proceso de moldurado.

Proceso de Moldurado

En la transformación secundaria de la madera, uno de los principales procesos que se identifican tiene que ver con la producción de molduras, los cuáles son productos destinados principalmente para la construcción de viviendas.

El producto en esencia es una escuadría de madera, la cual es sometida a un proceso de arranque de viruta, obteniéndose como resultado un perfil según sean las especificaciones de diseño. La acción de arrancamiento de viruta es logrado por una o mas frezas de corte, las cuales están montados en los respectivos cabezales de la máquina.

El número de cabezales que contiene la moldurera junto a la potencia del motor, define una serie de características de la máquina, como productividad, tiempos de setup, y los tipos de productos que son viables de confeccionar. Desde la perspectiva de operación de la máquina, cada vez que se ejecuta un cambio de moldura o perfil o que las especificaciones del producto no se logren por un desperfecto en algunas de las frezas de corte, es necesario efectuar un "setup " de la máquina con el correspondiente tiempo de detención. Desde el punto de vista de programación de la máquina (tipos de molduras) y de los criterios de programación (secuencia), los tiempos de "setup" adquieren una importancia fundamental en la productividad del sistema.

Análisis de los tiempos de setup.

El tiempo de setup de la máquina es dependiente de varios factores, los cuales se describen a continuación:

Del tipo de moldura que se desea fabricar.

Del número de cabezales que se debe intervenir

De la experiencia del operador en el cambio de los elementos de corte.

Del tiempo de calibración que se necesite.

El primer factor está relacionado al cambio de diseño de moldura que se quiere lograr en relación a la que se encuentre en proceso. Lo anterior va a condicionar el número de cambios que son necesarios de efectuar dentro de la máquina. El segundo factor tiene que ver con el número de cabezales que se deben intervenir para lograr la nueva moldura a ingresar en el proceso. El tercer factor corresponde a los procedimientos y experiencia que tenga el equipo de trabajo para efectuar los cambios respectivos. Es claro que si no se tiene un procedimiento estandarizado de intervención , los tiempos de setup entre los diferentes equipos presentará comportamiento aleatorio. El último factor se reacciona con la dificultad que tenga la calibración del equipo y con el grado de intervención y precisión del trabajo realizado. En este sentido dependiendo del nuevo diseño es posible la demanda de mayores cambios dentro de la máquina con los respectivos "costos" de calibración. En resumen, los dos primeros factores están relacionados con actividades que son eminentemente determinísticas, se pueden estandarizar por lo cual los tiempos de setup dependerán de la secuencia de la programación respectiva. En cambio en los segundos factores involucran actividades que en su esencia son de carácter estocástico y no dependen

de la secuencia de programación respectiva.

RESULTADOS

Con el fin de comparar el desempeño del algoritmo presentado en este estudio, se utilizaron dos metaheurísticas adicionales disponibles en la literatura para resolver el problema planteado. La primera de las técnicas utilizadas corresponde a Simulated Annealing Multiobjetivo, también conocido como MOSA, introducido por Ulungu et al. (1999). La segunda metaheurística utilizada es del ámbito de la programación evolutiva y se denomina Algoritmo Genético MultiObjetivo (MOGA) presentado por Fonseca & Fleming (1993).

Se consideró la realización de 20 experimentos los cuales consisten en la utilización de diferentes semillas en la ejecución de cada réplica, con el fin de evaluar la robustez del algoritmo. Como medida de desempeño se consideran dos elementos básicos del ámbito multiobjetivo, que son, el número de soluciones no dominadas generadas por cada algoritmo y la calidad de la frontera generada en términos de dominancia con respecto a las otras técnicas. Se utilizaron los mismos parámetros tanto para MOGA como para el algoritmo propuesto, esto con el fin de realizar la experimentación en las mismas condiciones. De esta manera, se utilizó una población de doscientos individuos y un total de 500 generaciones, con una probabilidad de cruzamiento del 100% y una de mutación del 2%. En el caso del algoritmo memético además fue necesario definir el parámetro de empeoramiento, el cual se estableció en base a experimentación previa considerando diferentes configuraciones del parámetro y evaluando su desempeño. Finalmente los mejores resultados se obtuvieron con un valor de veinte por ciento, es decir, cuando se aplica el operador de mejoramiento local en dirección del objetivo i, se permite avanzar en esta dirección hasta alcanzar un detrimento máximo de un veinte por ciento en el objetivo j. En el caso de MOSA, se consideró una temperatura inicial de cien grados con una velocidad de enfriamiento de 0.05 grados en cada etapa iterativa. Los resultados después de realizar las 20 corridas para cada experimento y heurística, se concentraron en una sola lista de soluciones no dominadas. La Tabla 1 muestra los resultados para ambos objetivos, las cuales representan la frontera de Pareto resultante. Estos valores están expresados en minutos y muestran claramente como el algoritmo propuesto domina al resto de las estrategias utilizadas.

Tabla 1: Resultados Expresados en Minutos

Alg. Propuesto		MOSA		MOGA	
Cmax	Tard. Total	Cmax	Tard. Total	Cmax	Tard. Total
2144	7893	2266	11034	2332	16270
2153	7399	2316	8117	2339	13569
2158	6153	2293	8591	2341	10140
2160	6034	2315	8435	2365	9583
2162	5999	2386	7402	2455	9555
2165	5799	2370	8062	2462	9243
2195	5780	2374	7773	2479	9154
2198	5693	2398	7213	2499	8976
2214	5621	2338	8074	2628	8883
				2639	8860
				2851	8570

En promedio es posible destacar que las soluciones generadas por el algoritmo propuesto superan a MOSA (el algoritmo que se acerca más a las soluciones del algoritmo memético) aproximadamente un 8% para el objetivo Cmax y un 35% para el objetivo tardanza total. Esta comparación representa una medida de calidad de la frontera resultante. Cabe mencionar que las tres metaheurísticas utilizadas fueron programadas especialmente para este trabajo utilizando el lenguaje C, razón por la cual no deberían existir diferencias en la calidad del código utilizado en cada caso. Los tiempos de ejecución de los algoritmos no superaron los 15 segundos para ninguna de las metahuerísticas, por lo cual no se hizo distinción en base a este aspecto de comparación por parecer irrelevante.

El número promedio de soluciones no dominadas obtenidas en las veinte réplicas de cada algoritmo se muestran en la Tabla 2.

Tabla 2: Número Promedio de Soluciones No-Dominadas

Metodología	Número Promedio de Soluciones no Dominadas
MOSA	6.8
MOGA	10.5
Algoritmo Memético Propuesto	9.3

En la Tabla 2 es posible ver que el número de soluciones no dominadas o en otras palabras el tamaño de la frontera de Pareto que genera cada algoritmo. Este parámetro muestra a MOGA como la mejor alternativa seguida por el algoritmo memético. Esta medida de rendimiento solo mide el tamaño de la frontera, pero no es capaz de definir si ésta es dominada totalmente o parcialmente por otras soluciones generadas por diferentes medios. En este sentido la medida de efectividad presentada por Hyun (1998), propone calcular el porcentaje de soluciones que corresponde a un algoritmo cuando la frontera de Pareto está compuesta por puntos generados por mas de un método. Este enfoque es muy simple y directo y permite evaluar la calidad de los procedimientos utilizados en un estudio. En este trabajo finalmente se concluyó que en ningún caso se observa que la mejor frontera de Pareto comparte soluciones de mas de un algoritmo, es más, existe una clara definición donde el algoritmo propuesto domina totalmente a los otros dos. Esto es mas fácil de observar mediante la Figura 1 la cual grafica los resultados de las veinte réplicas de cada algoritmo y que fueron presentados en la Tabla 1. El procedimiento consistió en agrupar las soluciones de todas las réplicas y eliminar las soluciones dominadas con el fin de estipular una frontera de Pareto. Es posible observar que el algoritmo memético propuesto supera en forma significativa a los otros algoritmos. Esto se expresa mediante la cercanía al origen de cada curva, debido a que ambos objetivos son de minimización.

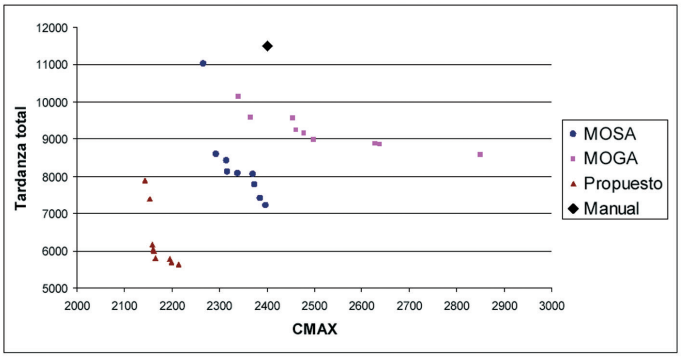

Figura 1: Fronteras de Pareto Resultantes

CONCLUSIONES

Este artículo presentó una nueva variante de algoritmo memético multiobjetivo introducido por los autores, el cual se probó su comportamiento en un problema real asociado a la problemático de programación de la producción del ámbito maderero. La diferencia principal que presenta esta variante de algoritmo memético consiste en el trabajo mediante poblaciones individuales para cada objetivo. Aunque el trabajo con poblaciones independientes no es totalmente nuevo, sí lo es la incorporación del mejoramiento local, típico del enfoque memético, pero en nuestro caso con la intención principal de lograr el compromiso o trade-off entre la soluciones, a diferencia de los enfoques de AM conocidos que utilizan la búsqueda local para realizar un mejoramiento de las soluciones encontradas en cada generación. El enfoque utilizado demostró la capacidad de encontrar fronteras de Pareto significativamente superiores en relación a las otras dos metahuerísticas utilizadas. Desde un punto de vista práctico al comparar los

resultados con respecto a la solución manual utilizada por la empresa para ese mismo problema se observa que el algoritmo propuesto disminuye en aproximadamente un 12% el objetivo Cmax, es decir es posible terminar que fabricar el lote completo en un 12% menos de tiempo, lo cual también se traduce directamente con la productividad del proceso. Con especto al objetivo tardanza total, fue posible disminuirlo en aproximadamente un 60% por ciento, es decir se logra disminuir significativamente el tiempo total de atraso con respecto a las fechas de entrega comprometidas a los clientes. Cabe mencionar que las tres metaheurísticas utilizadas superan significativamente los resultados obtenidos manualmente por la empresa, lo que indica la importancia de utilizar este tipo de técnicas en el campo aplicado.

REFERENCIAS

Buriol, L.; Franca, P. 2004. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics.* 10 (5): 483-506.

Deb, K.; Agrawal, S.; Pratab, A.; Meyarivan, T. 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, *In Proceedings of the Parallel Problem Solving from Nature VI Conference*, Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo and Hans-Paul Schwefel (editors), 849-858. Springer.

Drezner, Z. 2003. A new genetic algorithm for the quadratic assignment problem. Informs Journal on Computing 15 (3): 320-330

Fonseca, C.; Fleming, P. 1993. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *In Proceedings of the Fifth International Conference on Genetic Algorithms*, Stephanie Forrest, (editor) 416-423, San Mateo, California,1993. University of Illinois at Urbana-Champaign, Morgan Kau¨man Publishers.

Gen, M.; Cheng, R. 1997. *Genetic Algorithms and Engineering Design. Wiley.*

Goldberg, D., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Haupt, R.L., Haupt, S.E. 1998.*Practical Genetic Algorithms*. Wiley.

Horn, J.; Nafpliotis, N.; Goldberg, D. 1994. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. *In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence* 1 82-87, Piscataway, New Jersey, June 1994. IEEE Service Center.

Hyun, C.J.; Kim, Y.; Kin; Y.K. 1998. A genetic algorithm for multiple objective sequencing problems in mixed model assembly. *Computers and Operations Research* 25:675-690.

Ishibuchi, H.; Kaige, S. 2004. Implementation of Simple Multiobjective Memetic Algorithms and Its Application to Knapsack Problems. *International Journal of Hybrid Intelligent Systems* 1:22-35.

Ishibuchi, H.; Yoshida, T.; Murata, T. 2003. Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. *IEEE Transactions on Evolutionary Computation* 7(2):204-223.

Jaszkiewicz, A. 2004. A Comparative Study of Multiple-Objective Metaheuristics on the Bi-Objective Set Covering Problem and the Pareto Memetic Algorithm. *Annals of Operations Research* 131, (1-4): 135-158.

Srinivas, N.; Deb, K. 1994. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation* 2(3):221-248.

Ulungu, E.; Teghem, J.; Fortemps, P. 1999. MOSA Method: A Tool for Solving Multiobjective Combinatorial Optimization Problems. *Journal of Multi-Criteria Decision Analysis* 8(4):221-236

Zitzler, E.; Laumanns, M.; Thiele, L. 2002. SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *In EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou and T. Fogarty (eds.), 95-100, Athens, Greece, 2002.

Zitzler, E.; Thiele, L. 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 3(4):257-271.

Zydallis, J.; Van Veldhuizen, D.; Lamont G. 2001. A Statistical Comparison of Multiobjective Evolutionary Algorithms Including the MOMGA-II. *In First International Conference on Evolutionary Multi-Criterion Optimization,* Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, and David Corne, (editors), 226-240. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.