Subasta combinatoria para la programación dinámica en sistemas de fabricación distribuidos *Combinatorial auction for dynamic scheduling in distributed manufacturing systems*

José Alberto Araúzo¹, Ricardo del Olmo² y Juan José Laviós²

¹ Departamento de Organización de Empresas y CIM. Escuela de Ingenierías Industriales. Universidad de Valladolid. Pº del Cauce, 59, 47011, Valladolid, España.

² Departamento de Ingeniería Civil. Escuela Politécnica Superior (Edificio La Milanera). Universidad de Burgos. C/ Villadiego s/n, 09001, Burgos, España.
arauzo@eii.uva.es, rdelolmo@ubu.es, jjlavios@ubu.es

Fecha de recepción: 31-10-2012

Fecha de aceptación: 24-04-2013

Resumen: Los métodos de programación de operaciones tradicionales, estáticos y basados en arquitecturas centralizadas o jerárquicas, no son suficientemente flexibles para adaptarse al dinamismo y complejidad de los sistemas de fabricación actuales. Por ello, la investigación en técnicas dinámicas de programación y control *on-line* está creciendo rápidamente. En este artículo se presenta una técnica de programación dinámica *on-line* basada en modelos de mercado e implementada sobre un sistema multiagente. Esta propuesta, además de dinámica, es distribuida no jerárquica, lo que aporta al sistema las características deseadas.

Palabras claves: programación y control de la producción, programación *on-line*, sistemas de fabricación basados en agentes, subastas combinatorias.

Abstract: The traditional static scheduling methods, based on hierarchical and centralized architectures, are not flexible enough to self-adapt to the dynamism and complexity of today's manufacturing systems. For this reason, new proposals to improve the scheduling and control of agile manufacturing systems constantly appear. The auction based allocation methods as well as the software paradigm of multiagent systems, which offers new techniques to face complex unsolved problems, can help to find promising solutions in manufacturing systems.

Traditionally, scheduling problems have been solved offline by a centralized decision-maker that use a global optimisation model. We propose to include in the system several decision-makers modelled as agents instead. We consider two kinds of agents: order agents and machines agents. Each order agent represents a product that is characterized by its operations, precedence relationships and due date. The goal of each order agent is to find machines that can perform the required operations and hence completing successfully the order. Each production order creates its own schedule (local schedule). An auction mechanism ensures that local schedules are nearly compatible (several orders don't use the same machine at the same moment) and globally efficient.

Every agent in the system can communicate with other agents through the exchange of messages. The interaction mechanism is ruled by means of a combinatorial auction where a theoretical basis is provided for structuring message sequencing, bid evaluation, and price updating. Our research contributes to the auction technique in manufacturing scheduling and control in two basic aspects: (1) we apply the auction mechanism for a routing flexible environment (an operation can be performed in several machines with a differing efficiency), (2) we propose an implementation that can schedule online, updating real-time information: planning horizon, changes in orders, changes in machine availability and capabilities. We include explicitly the option of reallocate resources in real time when a new order arrives to the system.

In order to test the features of this approach we display some computational results. Preliminary results show efficient performance in dynamic scenarios, but there are still many matters to investigate. Future works will be devoted to test the proposed approach on more case studies or even on real cases. We can add complexity to the structure of the system, and we must improve some aspects of the auction mechanism such as convergence and stability.

Keywords: manufacturing programming and control, on-line scheduling, agent based manufacturing systems, combinatorial auctions.

1. Introducción

A pesar de la naturaleza dinámica y cambiante de los entornos productivos, gran parte de los estudios realizados sobre programación de operaciones se han centrado en la optimización de programas de modo estático, asumiendo que las diferencias en la ejecu-

ción respecto al programa no afectan a la calidad del mismo. Una de las causas de que así haya sido es la falta de información fiable que históricamente ha acompañado a la gestión de los sistemas productivos. Sin embargo, la llegada de los sistemas informáticos y redes de comunicación han hecho posible el seguimiento del estado de la planta, lo que permite

desarrollar sistemas de programación y control más eficientes en entornos dinámicos.

Los sistemas de producción actuales necesitan, por lo tanto, de sistemas de programación ágiles que sean capaces de adecuarse rápidamente a los cambios en las condiciones de partida mediante la modificación del programa de producción en tiempo de ejecución (Rajabinasab y Mansour 2010). El sistema debe detectar la existencia de perturbaciones y proponer, si es necesario, un nuevo programa que sustituya al anterior asegurando, de esta forma, su robustez o capacidad del sistema para mantener su efectividad en presencia de disrupciones, y facilitando su adaptabilidad o capacidad del sistema para acomodar los programas de producción a nuevas situaciones (Liu et al 2007).

Los enfoques tradicionales trataban de centralizar la información, así como la toma de decisiones que de ella se derivaba, a través de sistemas que, basados en modelos globales de la planta, estimaran el efecto de las diferentes decisiones sobre el conjunto. Pero, para que este sistema fuera eficiente era necesario que el modelo global fuese lo más exacto posible en todo momento, lo cual es muy difícil de conseguir. Además, modificar y adaptar los modelos globales es siempre una tarea compleja, que resta adaptabilidad y flexibilidad al conjunto.

Dada la rigidez y falta de robustez de los esquemas centralizados, las propuestas actuales para resolver este problema utilizan los enfoques distribuidos no jerárquicos (Tharumarajah 2003). Éstos, más acordes con la propia naturaleza de los sistemas de fabricación, permiten la incorporación en el proceso de decisión de objetivos locales, preferencias y restricciones de cada recurso o de cada orden, mejorando la rapidez de respuesta, la robustez, la capacidad de reconfiguración y la flexibilidad.

Pero las arquitecturas distribuidas presentan un problema: la necesidad de establecer mecanismos de coordinación para que el comportamiento global sea coherente y eficiente. Aunque en la literatura aparecen propuestas innovadoras, aún no se tienen soluciones universalmente aceptadas. Entre ellas se pueden citar los sistemas que implementan de forma distribuida los métodos de lanzamiento tradicionales (Babiceanu et al 2004, Hsieh 2008, Roulet-Dubonnet e Ystgaard 2011) o los sistemas *pull* (Ounnar y Pujo 2012). Son propuestas reactivas apropiadas para sistemas muy dinámicos, aunque pueden resultar poco eficaces. También se pueden encontrar sis-

temas de carácter predictivo en los que un coordinador central elabora programas que sirven de guía para la posterior toma de decisiones distribuida (Wonga et al 2006, Leitão y Restivo 2008). Estas últimas propuestas son más eficaces, pero al centralizar la información en el coordinador se pierde en parte las ventajas de los sistemas distribuidos: flexibilidad, robustez, adaptabilidad y rapidez de respuesta.

En este artículo se propone un método de programación dinámico y distribuido basado en subastas, implementado sobre un sistema multiagente (tecnología software que facilita el desarrollo de sistemas distribuidos), que opera en tiempo de ejecución, reflejando cualquier cambio que ocurra en el programa resultante. El sistema incluye enrutamiento flexible, posibilidad de reasignación de trabajos en tiempo de ejecución y actualización dinámica del horizonte de planificación. Esta propuesta pretende conservar las ventajas de los sistemas completamente distribuidos, pero mejorando la calidad de los programas conseguidos y, por lo tanto, su eficacia.

El resto del artículo se organiza como sigue. En el segundo apartado se presenta la aplicación de los sistemas basados en agentes en el ámbito de la producción y, a continuación, en el apartado tres, la programación mediante subastas. En el apartado cuatro se desarrolla el modelo de subasta. Después, en el quinto apartado se exponen los resultados obtenidos, para finalizar con unas conclusiones.

2. Sistemas basados en agentes en el ámbito de la producción

La aplicación de los sistemas basados en agentes en el ámbito de la producción pretende explotar los conceptos, métodos y técnicas introducidas desde el paradigma informático de sistemas multiagente, con el fin de desarrollar sistemas de fabricación distribuidos más competitivos.

Desde el punto de vista de este paradigma, un agente es una entidad software autónoma que es capaz de actuar en un entorno. Posee comportamientos proactivos (orientados por objetivos) y reactivos (reacciona ante estímulos) (Wooldridge 2009). Además, dado que en el entorno pueden existir más agentes, éstos deberán ser capaces de interactuar entre sí. Estas características ofrecen a los desarrolladores de software la posibilidad de crear comunidades de agentes (sistema multiagente), que colaboran para con-

seguir una meta global. De esta forma, el paradigma multiagente proporciona los métodos para diseñar e implementar sistemas distribuidos complejos.

Este paradigma se ha aplicado a bastantes problemas relacionados con la gestión empresarial. Por ejemplo, gestión de la cadena de suministros (Moyaux et al 2006), integración (Kishore et al 2006), programación de proyectos (Araúzo et al 2009). Un problema muy interesante es la programación y control de plantas de producción (Marík et al 2011) para el que, debido a su complejidad y su naturaleza distribuida, inestable, abierta y dinámica, los sistemas multiagente son una tecnología muy prometedora.

En los sistemas de control de planta basados en agentes, un conjunto de agentes modelan y controlan diferentes elementos concretos de la planta (máquinas, órdenes de producción, vehículos, robots, etc.). Como cada agente solo posee un modelo parcial de la planta (conocimiento limitado), solo se podrá alcanzar el objetivo global del sistema si los agentes interactúan, acordando planes conjuntos mutuamente beneficiosos, para posteriormente intentar ejecutar dichos planes.

Aunque hasta el momento se han propuesto bastantes arquitecturas multiagente para el control de sistemas de fabricación, el enfoque más aceptado propone dos tipos de agentes básicos (Araúzo et al 2003): un conjunto de agentes que modelan cada recurso —generalmente máquinas—, y un conjunto de agentes que modelan cada orden de producción. Los agentes que modelan las máquinas tienen conocimiento sobre las operaciones que pueden ejecutar, siendo su objetivo conseguir trabajo. Los agentes que modelan las órdenes tienen como objetivo conseguir su fabricación en el plazo deseado, y su conocimiento versa sobre su proceso de producción (operaciones y precedencias) y plazo. Estos agentes deberán contactar con los agentes que modelan las máquinas, y acordar planes conjuntos con los que conseguir los objetivos planteados.

3. Programación basada en subastas para sistemas de control de planta basados en agentes

Muchos problemas prácticos de optimización combinatoria pueden ser considerados de forma genérica como problemas combinatorios de asignación de recursos (CAP). En ellos, un conjunto de recur-

sos no divisibles debe ser asignado a diferentes agentes, de forma que la utilidad global del reparto sea máxima. La relación de este tipo de problemas con la programación de talleres es inmediata (Welman et al 2001). Basta con considerar que las órdenes de producción son agentes que deben repartirse las unidades de tiempo (*slots*) disponibles en las diferentes máquinas con el objeto de poder ser fabricadas. La utilidad que cada orden de producción obtendrá por cada conjunto de *slots* adquiridos, será función de sus operaciones, duraciones, relaciones de precedencia y plazos, etc. (Figura 1).

Figura 1
Programación como un problema de asignación de recursos

La ventaja de esta formulación, además de adaptarse perfectamente con el enfoque multiagente del apartado anterior, es la similitud que posee con los mercados, donde un número de agentes (órdenes) intentan maximizar su utilidad adquiriendo bienes (*slots*) a unos vendedores que intentan maximizar su beneficio (máquinas). Esta analogía permitirá introducir en el problema otras variables, imprescindibles en todo mercado, que pueden ayudar a establecer nuevos procedimientos de resolución para los CAP y por extensión para la programación de sistemas de fabricación. Dichas variables serán los precios de los bienes.

Los procedimientos de subasta persiguen la obtención de los precios (llamados de equilibrio) para los que la maximización de los excedentes individuales de cada agente da lugar a asignaciones globalmente óptimas. De esta forma, si se encontraran dichos precios, el problema se podría resolver maximizando de forma independiente los excedentes de cada agente (objetivo particular de cada agente). Las subastas describen los mecanismos, controlados por un subastador, para determinar la asignación del bien intercambiado y los precios en función de las pujas que realizan los participantes (Krishna 2009).

3.1. Tipos de subastas para la programación de operaciones

En los problemas de programación de operaciones, los ítems subastados presentan complementariedades, de modo que la valoración de un determinado slot depende de si va acompañado o no de otro conjunto de slots. Esto es debido a las restricciones de precedencia y a la duración de las operaciones de las órdenes, que hace que un único slot no aporte ninguna utilidad si no va acompañado de otros que garanticen la fabricación de la orden completa. Estas complementariedades hacen que la existencia de precios de equilibrio no esté garantizada, lo que complica el problema (Welman et al 2001). Para solucionarlo se utilizan subastas combinatorias (Cramton et al 2006). En ellas, a diferencia de las subastas simples, los participantes pujan a la vez por un conjunto de bienes complementarios entre sí.

Hay distintos tipos de subasta combinatoria: (1) las de una ronda en la que los participantes envían una sola vez las distintas valoraciones de las posibles combinaciones de bienes y el subastador asigna los bienes de modo que se maximiza el objetivo global, y (2) las iterativas en las que los precios se fijan después de múltiples rondas. Estas últimas tienen varias ventajas. En primer lugar, los participantes no necesitan realizar pujas sobre todo el conjunto de posibles combinaciones. En segundo lugar, los participantes revelan en cada iteración sus preferencias, a través de los precios que se generan. Por último, las subastas iterativas son adecuadas en entornos dinámicos, donde los participantes entran y salen en cualquier momento.

Así mismo, hay dos métodos para realizar las pujas en subastas iterativas. En un tipo, los participantes envían el precio que están dispuestos a pagar por las posibles combinaciones de bienes y el subastador asigna los bienes a cada participante en base a esas valoraciones (subastas con fijación de cantidad). Aplicaciones de este tipo de subastas en la programación de talleres pueden verse en Tiwari y Allada (2004), y en la programación de proyectos en Lee et al (2003).

En el segundo tipo de subastas (subastas de fijación de precios), los participantes envían como puja el conjunto de bienes que estarían dispuestos a adquirir a los precios ofrecidos por el subastador. En cada iteración, el subastador actualiza los precios en función de la demanda. Pueden encontrarse aplicaciones de estas subastas a la programación *job shop* en

Liu et al (2007) o en Jeong y Yim (2009), y para la programación de proyectos en Araúzo et al (2009).

4. Modelo de subasta propuesto

El sistema de programación propuesto en este trabajo está basado en subastas combinatorias iterativas de fijación de precios. Se aplicará a un entorno de fabricación *job shop*, dinámico y con enrutamiento flexible, donde cada operación puede ser fabricada en más de una máquina con diferentes duraciones.

4.1. Entorno de fabricación

Se debe fabricar un conjunto de órdenes, dentro de un plazo, cada una con una importancia relativa diferente. Las órdenes aparecen dinámicamente en tiempo de ejecución, desconociéndose el momento de llegada de cada una.

Una orden se compone de un conjunto de operaciones cuya ejecución está limitada por restricciones de precedencia. Cada operación se puede realizar en una o varias máquinas con tiempo de ejecución diferente, lo que da lugar a diferentes rutas de fabricación posibles. Se supone que cada máquina solo puede procesar una operación a la vez, y que una vez comenzada no puede interrumpirse (modo non-pre-emptive). No se considera el transporte de material entre las máquinas, ni tiempos de preparación dependientes de la secuencia. Se supone, además, tamaño infinito de colas.

Estas hipótesis no limitan la aplicabilidad del procedimiento ya que podría extenderse a problemas donde no se tengan en cuenta estas simplificaciones. Las tareas de transporte podrían ser fácilmente introducidas en el modelo como actividades de las órdenes que usan recursos. Lo mismo sucede con los tamaños de cola finitos que pueden tratarse como recursos usados por las órdenes en sus esperas entre operaciones. Además, hay trabajos, como Liu (2000), que aportan ideas sobre cómo considerar tiempos de preparación dependientes de la secuencia. Aun así, no se considerarán estas situaciones para no introducir excesiva complejidad en el problema y facilitar el análisis de resultados.

Para medir la eficacia global del sistema se utilizará la media ponderada de los retrasos cuadráticos ($\overline{T^2}$) de todas las órdenes que se han lanzado durante el periodo de tiempo considerado:

$$\overline{T^2} = \frac{\sum_{i=1}^N w_i \cdot T_i^2}{\sum_{i=1}^N w_i} \quad [1]$$

$T_i = \max(0, \max_j(c_{ij}) - D_i) \rightarrow$ Retraso de la orden i

$w_i \rightarrow$ Peso de la orden i

$D_i \rightarrow$ Plazo de entrega de la orden i

$c_{ij} \rightarrow$ Fecha de conclusión de la operación j de la orden i

$N \rightarrow$ Número de órdenes

Se ha optado por esta expresión porque penaliza más los grandes incumplimientos de plazo.

4.2. Diseño de la subasta dinámica

En las subastas iterativas de fijación de precios, el subastador asigna un precio a cada uno de los artículos a subastar. Cada participante envía como puja la combinación de elementos que pretende adquirir a unos precios determinados, de forma que se maximice su utilidad. El subastador va adaptando los precios en cada iteración en base a la demanda de cada artículo. Dado que en este tipo de subastas el subastador juega un papel fundamental, es necesario introducirlo como un agente en el sistema junto con las órdenes y las máquinas. En cada iteración (ronda) la subasta se desarrolla según los siguientes pasos.

Paso 1 (Actualización de precios). El subastador solicita a las máquinas los *slots* ocupados por las operaciones en curso y fija el precio M (valor suficientemente grande) para esos *slots* ocupados. Para el resto, los precios se fijan en función de la demanda de esos *slots* por parte de las órdenes en la ronda anterior. Si la demanda supera a uno, el precio se aumenta en función del exceso de demanda, si es uno se mantiene igual, y si es cero se reduce:

$$p_{mt}^{(0)} = 0$$

$$p_{mt}^{(n)} = \max(0, p_{mt}^{(n-1)} + a^n \cdot (d_{mt}^{(n-1)} - 1)) \quad [2]$$

$p_{mt}^{(n)} \rightarrow$ Precio del *slot* (t) de la máquina (m) en la ronda (n).

$d_{mt}^{(n-1)} \rightarrow$ Demanda del *slot* (t) de la máquina (m) en la ronda ($n-1$):
número de órdenes que han incluido en ese *slot* en sus programas

$a^n \rightarrow$ Paso de actualización en la ronda (n)

El aumento o disminución de los precios está regulado por un factor α que se va reduciendo a medi-

da que la subasta converge. Para calcularlo se ha utilizado el método derivado de la relajación lagrangiana que se puede encontrar en Baptiste et al (2008).

Para cada máquina, los precios se actualizan, desde el *slot* inmediatamente posterior al instante actual, hasta el más lejano que esté ocupado o que haya sido solicitado por alguna orden. A partir de ahí se interpreta que todos los precios son cero, sin indicarlo expresamente. Esto permite considerar el horizonte de planificación que sea necesario, aumentándose a medida que las órdenes lo van necesitando.

Paso 2. Las órdenes proponen programas locales con los precios propuestos por el subastador. Un programa local es el paquete de *slots* que la orden considera adecuado para ejecutar sus operaciones. De todos los posibles paquetes de *slots*, la orden selecciona aquel que maximiza su utilidad, o dicho de otra forma, minimiza el coste de ejecución de las operaciones, incluyendo en este coste, el coste de adquisición de los *slots* al precio propuesto por el subastador y el coste del posible retraso del programa sobre el plazo de entrega de la orden:

$$C_i^n(s_i) = \sum_{ht \in S_i} p_{mt}^{(n)} + w_i \cdot T_i(S_i)^2 \quad [3]$$

$S_i \rightarrow$ Paquete de *slots* seleccionados por la orden (i)

$C_i^n(S_i) \rightarrow$ Coste del paquete de *slots* seleccionados por (i)

$p_{mt}^{(n)} \rightarrow$ Precio del *slot* (t) de la máquina (m)

$T_i(S_i) \rightarrow$ Retraso de la orden (i) suponiendo el programa derivado del paquete

Para seleccionar el paquete de *slots* de mínimo coste las órdenes utilizan un algoritmo enumerativo de programación dinámica, donde se consideran todos los posibles conjuntos de *slots* (Tang et al 2007). Dado que el horizonte de planificación no está definido de antemano ya que depende del número de órdenes presentes en el sistema y de sus necesidades, cada orden calculará el horizonte de planificación (número de *slots* a considerar):

$$H_i = \max_m (H_m) + MH_i \quad [4]$$

$H_i \rightarrow$ N.º de *slots* considerados por la máquina m (aquellos para los que el subastador a propuesto precios)

$MH_i \rightarrow$ Máximo número de *slots* que necesita la orden i suponiendo que las operaciones se programarán seguidas (sin esperas entre ellas)

Esto garantiza que se seleccionará el conjunto de slots de mínimo coste, acotando las necesidades de computación.

Paso 3. El subastador recibe los programas locales propuestos por las órdenes y construye un programa global factible. Aunque a medida que la subasta converge y los precios se van estabilizando, los programas locales serán más compatibles (no hay slots utilizados por dos o más programas locales), siempre se puede dar algún conflicto, por lo que es necesario construir un programa global factible a partir de los programas locales. Para elaborar el programa global se utiliza una técnica constructiva basada en el algoritmo de Giffer y Thompson (1960), utilizando la siguiente regla heurística: tendrán preferencia las operaciones que antes finalicen según el programa local de cada orden y en caso de empate la regla de *minimun-slack* por tarea pendiente. Aunque existen más alternativas para la elaboración de los programas globales, como las técnicas tradicionales de lanzamiento sin esperas o el algoritmo propuesto por Hoitomt et al. (1993), se ha optado por el método de Giffer y Thompson porque es sencillo y capaz de generar todos los programas activos posibles.

Un vez finalizado el paso 3, el subastador comienza una nueva ronda de la subasta recalculando los precios según el paso 1 en función de la demanda ori-

ginada por los programas locales que las órdenes han propuesto en el paso 2 de la iteración anterior. Este proceso se repite continuamente y, con el tiempo, los precios convergen hacia una situación de equilibrio. Estos precios contienen información de la importancia que tienen los diferentes slots de las máquinas en la consecución de los objetivos globales. A medida que se alcanza el equilibrio, los programas locales elaborados por las órdenes serán más compatibles globalmente, y los programas globales derivados, generalmente más eficientes.

4.3. Ejemplo

Para ilustrar el procedimiento se desarrolla a continuación un ejemplo muy simple, con dos órdenes y dos máquinas (M1 y M2), donde se muestra cada paso de la subasta en varias rondas.

En la Figura 2 se recogen los pasos 1, 2 y 3 de la primera ronda (de inicialización) de la subasta. Para comenzar, en el paso 1, el subastador fija precio cero para todos los slots de todas las máquinas. En el paso 2, las órdenes proponen los programas locales que para estos precios consisten en la ejecución consecutiva de las operaciones. La unión de programas locales no es factible, por lo que el subastador, cuando los recibe, elabora un programa global eliminando los conflictos que haya (paso 3).

Figura 2

Pasos 1, 2 y 3 de la ronda de inicialización de la subasta para un ejemplo ilustrativo con dos órdenes y dos máquinas (M1 y M2)

En la Figura 3 se muestran los pasos 1, 2 y 3 de la segunda ronda de la subasta. Dado que en la ronda de inicialización, el cuarto slot de la máquina M2 ha sido solicitado por las dos órdenes (para la Op12 por la orden 1 y para la Op22 por la orden 2), en el paso 1 de esta segunda ronda el subastador aumenta su precio. Esto hace que en el paso 2, cuando las órdenes realicen sus programas locales, la orden 2 que tiene un plazo más amplio, evite dicho slot, mientras que la orden 1, que tiene un plazo más ajustado, lo siga asignando a sus operaciones para poder concluir a tiempo. Esto hace que los programas locales sean globalmente factibles, y que el programa resultante de su unión sea, además, óptimo. Aunque para este ejemplo, dada su sencillez se podían haber obtenido los mismos resultados programando dinámicamente mediante reglas de prioridad, esto no sucede en ejemplos más complejos. Además, en ejemplos más complejos son necesarias más rondas de la subasta para conseguir buenos programas.

En la Figura 4 se muestran los pasos 1, 2 y 3 de la segunda ronda de la subasta para el ejemplo ilustrativo en $t = 2$. En ese instante aparece una nueva orden con una sola operación que se puede ejecutar en cualquiera de las dos máquinas, con una duración de 4 en M1 y de 3 en M2. Esta nueva orden prefiere programar su operación en la M1, aunque sea menos eficiente y eso implique un pequeño retraso, evitando

los precios elevados que tienen los slots necesarios de la M2. Si el peso de la orden 3 hubiera sido mucho mayor que los pesos de las otras dos órdenes, la orden 3 podría haber programado su operación en la máquina M2 para evitar el retraso, con lo que, en rondas posteriores, los precios de estos slots aumentarían obligando a las órdenes 1 y 2 a retrasar sus operaciones. Este caso ilustra como los precios van reflejando las nuevas situaciones, permitiendo a las órdenes tomar decisiones locales globalmente eficientes.

5. Resultados computacionales

Para testar el sistema se han realizado pruebas donde se compara el desempeño de este método respecto a los basados en reglas de lanzamiento, típicamente aplicados en entorno dinámicos. Con objeto de mostrar el correcto funcionamiento del procedimiento, primero se han realizado pruebas sobre 12 problemas estáticos tipo *job shop* con cinco máquinas, que han sido generados aleatoriamente con duraciones distribuidas uniformemente entre 30 y 120 segundos. De los 12 problemas, 4 de ellos tienen 5 órdenes, hay otros 4 con 7 órdenes, y 4 más con 10 órdenes. Los pesos se han generado aleatoriamente entre 1 y 5, y los plazos de entrega, entre 5 y 7 minutos para los problemas pequeños (de 5 órdenes),

Figura 3

Pasos 1, 2 y 3 de la segunda ronda de la subasta para el ejemplo ilustrativo de la Figura 2.

entre 5 y 10 para los medianos y entre 5 y 15 para los grandes.

La Figura 5 muestra el valor de la función objetivo —retraso cuadrático ponderado— conseguido con el procedimiento propuesto frente al obtenido con un procedimiento de reglas de lanzamiento clásico al que se le aplica la heurística de *minimun slack* por operación —dentro de las operaciones en cola de máquina se programa aquella que menos holgura tiene por operación respecto a su fecha de entrega—. Los resultados se muestran en porcentaje de la solución obtenida, frente a la obtenida mediante las reglas de lanzamiento. Como se puede apreciar, la mejora que se produce aplicando el método propuesto es substancial.

Figura 5

Valor de la función objetivo conseguido con el método de programación propuesto frente al conseguido con regla de lanzamiento de *minimun slack* por operación, para los 12 problemas propuestos. El primer número que identifica el escenario indica la cantidad de órdenes, el segundo es un número correlativo

Después de las pruebas estáticas se han realizado pruebas dinámicas sobre cada problema, añadiendo órdenes en tiempo de ejecución que se suman a las ya lanzadas en el modo estático. En la Figura 6 se representa el resultado obtenido con el mecanismo de subasta, comparado con el obtenido con la regla de lanzamiento, en 11 problemas derivados del problema 7/1, con diferente número de órdenes añadidas en tiempo de ejecución (las órdenes se añaden cada 60 segundos). Se puede observar como la eficacia del procedimiento sigue siendo mejor frente al método de lanzamiento, incluso en condiciones de alto dinamismo.

En los experimentos realizados, el tiempo de reacción ante la entrada de una nueva orden es suficientemente pequeño para garantizar la reactividad del sistema. El tiempo que transcurre desde que entra una nueva orden hasta que se estabilizan los precios está entre 5 y 15 segundos. Durante este tiempo se

realizan entre 25 y 100 rondas de subasta, dependiendo del impacto que tenga la nueva orden en el programa existente. Se debe destacar que aunque los precios no se hayan estabilizado, los programas resultantes pueden ser de gran calidad, ya que la nueva orden intentará evitar los *slots* con precios elevados, aquellos que son importantes para el cumplimiento de plazos.

Figura 6

Comparación de resultados obtenidos con el método propuesto respecto a los obtenidos con el método de lanzamiento con *minimun slack* por operación para 11 escenarios dinámicos en los que se ha ido aumentando el número de órdenes que entran en tiempo de ejecución en el problema 7/1

La parte del algoritmo con más exigencia de cómputo es el cálculo de los programas locales por parte de cada orden. Es un algoritmo que crece de forma no polinómica al aumentar el número de operaciones de la orden. Pero esto no supone un problema insoslayable en entornos reales, en los que el número de máquinas, órdenes y operaciones por orden, puede ser elevado. Una de las ventajas del método propuesto es que el cálculo de programas locales se puede distribuir sobre diferentes unidades de cálculo, cada una de ellas dedicada al programa local de una o varias órdenes. Ello permitiría el tratamiento de problemas de tamaños bastante mayores.

6. Conclusiones y extensiones

Con objeto de mejorar el desempeño de los sistemas de fabricación actuales, donde el dinamismo es un factor clave, se proponen sistemas de control distribuidos con arquitecturas no jerárquicas. Especial interés suscitan, dentro de este campo, los sistemas de fabricación basados en agentes, sobre todo en los aspectos relacionados con la programación y control de plantas de fabricación. Desde este paradigma se propone distribuir el sistema de control de la fabricación en dos tipos de agentes: agentes asociados a las órdenes de producción y agentes asociados a los

recursos. Esta distribución permite establecer una analogía con los mercados donde las órdenes pujan por los servicios que ofrecen las máquinas.

Siguiendo el esquema anterior, hemos desarrollado un sistema de programación basado en subastas que es capaz de reflejar en todo momento los cambios que aparecen en la planta de producción. Las pruebas realizadas con este mecanismo en un escenario muy dinámico muestran claramente como el comportamiento es bastante más eficiente que el conseguido con las reglas de prioridad tradicionalmente aplicadas en estos entornos.

La experiencia acumulada durante el desarrollo de nuestro trabajo nos permite proponer algunas líneas de investigación encaminadas a la descentralización de las partes más centralizadas del algoritmo, lo que lo adecuaría aún más al paradigma distribuido para el que ha sido diseñado. Desde ese punto de vista, sería interesante distribuir el cálculo de precios entre los agentes máquina proponiendo otros métodos de actualización de precios, y además, estudiar el comportamiento del algoritmo ejecutándose de forma asíncrona.

Bibliografía

- ARAÚZO, J.A., DE BENITO, J.J. y DEL OLMO, R. (2003). «El control de planta: una aproximación basada en agentes». *Actas del V Congreso de Ingeniería de Organización*.
- ARAÚZO, J.A., PAJARES, J., LÓPEZ-PAREDES, A. y PAVÓN, J. (2009). «Agent-based modeling and simulation of multi-project scheduling». *Proceedings of the Multi-Agent Logics, Languages, and Organizations Federated Workshops (MALLOW 2009)*.
- BABICEANU, R.F., CHEN, F.F., STURGES R.H. (2004). «Framework for the control of automated material handling systems using the holonic manufacturing approach». *International Journal of Production Research*, 42(17), pp. 3551-3564.
- BAPTISTE, P., FLAMINI, M. y SOURD, F. (2008). «Lagrangian bounds for just-in-time job-shop scheduling». *Computers and Operations Research*, 35(3), pp. 906-915.
- CRAMTON, P., SHOHAM, Y. y STEINBERG, R. (2006). *Combinatorial auctions*. Cambridge: MIT Press.
- GIFFER, B. y THOMPSON, G. L. (1960). «Algorithms for solving production-scheduling problems». *Operations Research*, 8(4), pp. 487-503.
- HOITOMT, D.J., LUH, P.B. y PATTIPATI, K.R., (1993). «A practical approach to job-shop scheduling problems». *IEEE Transactions on Robotics and Automation*, 9(1), pp. 1-13.
- HSIEH, F.S. (2008). «Holarchy formation and optimization in holonic manufacturing systems with contract net». *Automatica*, 44(4), pp. 959-970.
- JEONG, I. y YIM, S.B. (2009). «A job shop distributed scheduling based on Lagrangian relaxation to minimise total completion time». *International Journal of Production Research*, 47(24), pp. 6783-6805.
- KISHORE, R., ZHANG, H. y RAMESH, R. (2006). «Enterprise integration using the agent paradigm: foundations of multi-agent-based integrative business information systems». *Decision Support Systems*, 42, pp. 48-78.
- KRISHNA, V. (2009). *Auction theory* (2 ed.). San Diego: Academic Press.
- LEE, Y.H., KUMARA, S.R.T. y CHATTERJEE, K. (2003). «Multiagent based dynamic resource scheduling for distributed multiple projects using a market mechanism». *Journal of Intelligent Manufacturing*, 14(5), pp. 471-484.
- LEITÃO, P., RESTIVO, F., (2008). «A holonic approach to dynamic manufacturing scheduling». *Robotics and Computer-Integrated Manufacturing*, 24(5), pp. 625-634.
- LIU C.Y. (2000) «Scheduling flexible flow shops with sequence-dependent setup effects». *IEEE Transactions on Robotics and Automation*, 16(4), pp. 408-419.
- LIU, N., ABDELRAHMAN, M.A. y RAMASWAMY, S. (2007). «A complete multiagent framework for robust and adaptable dynamic job shop scheduling». *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, 37(5), pp. 904-916.
- MARÍK, V., VRBA, P. y LEITÃO, P. (eds.) (2011). *Holonic and multi-agent systems for manufacturing*. Berlin: Springer.
- MOYAUX, T., CHAIB-DRAA, B. y D'AMOURS, S. (2006). «Supply chain management and multiagent systems: An overview». En B. Chab-draa y J. Müller (eds.), *Multiagent based Supply Chain Management*, pp. 1-27. Berlin: Springer.
- OUNNAR, F., PUJO, P. (2012). «Pull control for job shop: holonic manufacturing system approach using multicriteria decision-making». *Journal of Intelligent Manufacturing*, 23(1), pp. 141-153.
- RAJABINASAB, A. y MANSOUR, S. (2010). «Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach». *The International Journal of Advanced Manufacturing Technology*, 54(9-12), pp. 1091-1107.
- ROULET-DUBONNET, O. y YSTGAARD, P. (2011). «An application of the holonic manufacturing system to a flexible assembly cell». *Lecture Notes in Computer Science*, 6867, pp. 29-38.
- TANG, L., XUAN, H. y LIU, J. (2007). «Hybrid backward and forward dynamic programming based Lagrangian

- relaxation for single machine scheduling». *Computers and Operations Research*, 34(9), pp. 2625-2636.
- THARUMARAJAH, A. (2003). «From fractals and bionics to holonics». En S.M. Deen (ed.), *Agent-Based Manufacturing: Advances in the Holonic Approach*, pp. 11-30. Berlin: Springer.
- TIWARI, M.K. y ALLADA, V. (2004). «Solving the machine-loading problem in a flexible manufacturing system using a combinatorial auction-based approach». *International Journal of Production Research*, 42(9), pp. 1879-1893.
- WELMAN, M.P., WALSH, W.E., WURMAN, P.R. y MACKIE-MASON, J.K. (2001). «Auction protocols for decentralized scheduling». *Games and Economic Behaviour*, 35, pp. 271-303.
- WONGA, T.N., LEUNGA, C.W., MAK, K.L. y FUNGB, R.Y.K. (2006). «Dynamic shopfloor scheduling in multi-agent manufacturing systems». *Expert Systems with Applications*, 31(3), pp. 486-494.
- WOOLDRIDGE, M. (2009). *An introduction to multi-agent systems*. Chichester: Wiley.