Universidade Federal de Santa Catarina | www.periodicos.ufsc.br ISSNe 2175-8018
IJIE
IBEROAMERICAN JOURNAL OF INDUSTRIAL ENGINEERING
REVISTA IBEROAMERICANA DE ENGENHARIA INDUSTRIAL
REVISTA IBEROAMERICANA DE INGENIARÍA INDUSTRIAL

PROGRAMACIÓN DE LA PRODUCCIÓN A CORTO PLAZO Y DE TAREAS DE MANTENIMIENTO PREVENTIVO EN AMBIENTES *JOB SHOP* FLEXIBLES

Mauricio Daniel Sirolla[1]

Juan Matias Novas[2]

Gabriela Patricia Henning[3]

RESUMEN: Se aborda el problema de *scheduling* predictivo en plantas industriales de tipo *Job Shop* Flexible para el que se ha desarrollado un modelo basado en programación con restricciones (*Constraint Programming* – CP) que permite obtener una agenda eficiente para un conjunto de partes o *Jobs* conocido de antemano. El modelo considera las características de este tipo de ambiente industrial: recetas de manufactura específicas para cada *Job*, unidades multipropósito y disímiles para llevar a cabo cada operación, tiempos de alistamiento de equipos y de disponibilidad inicial de las máquinas, etc. Además, la formulación considera la necesidad de ejecutar tareas de mantenimiento preventivo en cada equipo. El modelo se ha verificado y validado utilizando diferentes ejemplos de tamaño medio disponibles en la bibliografía, hallándose soluciones de muy buena calidad en bajos tiempos de cómputo, lo que permite inferir las bondades del modelo.

Palabras claves: *Scheduling* predictivo. *Job Shop* Flexible. Programación con restricciones.

1 INTRODUCCIÓN

Los ambientes industriales de tipo *Job Shop Flexible* (*Flexible Job Shop* – FJS) generalizan los entornos *Job Shop* (JS) simples, ya que cuentan con mayores alternativas para el procesamiento de las partes demandadas (*Jobs*). En un FJS cada operación de manufactura puede realizarse en un equipo perteneciente a un subconjunto de máquinas alternativas, lo cual dificulta el problema de asignación de tareas a unidades. En general, las máquinas que componen un ambiente FJS demandan actividades de mantenimiento preventivo.

[1] Ingeniero Industrial, INTEC (Universidad Nacional del Litoral, CONICET), Güemes 3450, Santa Fe, 3000, Argentina. E-mail: msirolla@intec.unl.edu.ar

[2] Dr. en Ingeniería, Mención en Ingeniería en Sistemas de Información CIEM (Universidad Nacional de Córdoba, CONICET), Medina Allende s/n, Córdoba, 5000, Argentina. E-mail: jmnovas@famaf.unc.edu.ar

[3] Dra. en Ingeniería Química, INTEC (Universidad Nacional del Litoral, CONICET), Güemes 3450, Santa Fe, 3000, Argentina. E-mail: ghenning@intec.unl.edu.ar

Éstas se ejecutan con periodicidad con el objeto de permitir que los equipos operen de forma adecuada. Durante las tareas de mantenimiento las máquinas quedan inhabilitadas para realizar operaciones de manufactura. Por ello, es necesario programar en conjunto las tareas de manufactura y de mantenimiento, incluyendo ambos tipos de actividades en la agenda de trabajo o *schedule*.

En el presente trabajo se propone el desarrollo de una formulación basada en programación con restricciones (*Constraint Programming* – CP), que contemple la planificación conjunta de las operaciones de manufactura y las tareas de mantenimiento preventivo en ambientes FJS. En la sección 2 se describe el problema abordado y las hipótesis adoptadas. En las secciones 3, 4 y 5 se presenta la formulación desarrollada. En la siguiente sección se reportan y discuten los resultados obtenidos al resolver distintos casos de estudio. Finalmente, se presentan las conclusiones y trabajos futuros.

2 DESCRIPCIÓN DEL PROBLEMA

En la categoría de problemas clásicos de *scheduling* (LEUNG, 2004) se enmarca el problema de *scheduling* predictivo de ambientes FJS. Este último ha sido abordado mediante diversas metodologías (GAO; SUN; GEN, 2008; KARIMI; RAHMATI; ZANDIEH, 2012; WANG; YU, 2010; XIE et al., 2009) como los algoritmos combinatorios, métodos heurísticos y metaheurísticos, búsqueda tabú, colonia de hormigas, etc. Una de las dificultades de estas propuestas es su escasa capacidad para incorporar características adicionales propias de los ambientes industriales. Por el contrario, CP permite incluirlas y resolver problemas combinatorios mediante el desarrollo de modelos expresivos y extensibles. Diversos autores consideran que este enfoque es apropiado para abordar problemas de *scheduling* (BRAILSFORD; POTTS; SMITH, 1999).

La obtención de una agenda de trabajo en respuesta al problema de *scheduling* predictivo en ambientes FJS es una actividad compleja, no sólo desde el punto de vista práctico sino también desde la perspectiva matemática, pues se trata de un problema combinatorio de tipo *NP-hard* (GAREY; JOHNSON; SETHI, 1976). Al agregar flexibilidad en la asignación de las operaciones a las máquinas, la complejidad del problema es mayor.

Los ambientes FJS se conforman de un conjunto de máquinas encargadas de llevar a cabo las operaciones demandadas por las partes. Dichas máquinas tienen capacidad de realizar diferentes tareas (unidades multipropósito), pero ejecutan una única operación a la vez. Cada parte o *Job j* requiere un conjunto de operaciones O_j *y* cada una de éstas demanda un tiempo

que depende del equipo que la lleva a cabo. En este trabajo se considera que las máquinas del ambiente FJS se someten a tareas de mantenimiento durante el período de planificación.

Así, la obtención de la agenda de trabajo implica: (i) asignar máquinas a las operaciones requeridas por cada *Job*, (ii) secuenciar las tareas asignadas a cada máquina, (iii) especificar el tiempo de inicio de cada operación de manufactura y (iv) establecer el comienzo de cada tarea de mantenimiento preventivo.

A efectos de obtener agendas de buena calidad se han considerado diferentes medidas de desempeño a optimizar: (i) *Makespan*: tiempo de finalización de la última operación requerida por el último *Job* en concluir, (ii) Carga total de las máquinas: suma de los tiempos requeridos por todas las tareas asignadas a ellas, (iii) Carga máxima de las máquinas: suma de los tiempos de operación de las tareas asignadas al equipo con mayor ocupación. Además, se ha contemplado la optimización conjunta de estos criterios mediante la minimización de una función multiobjetivo.

El presente trabajo contempla las siguientes características del ambiente FJS: (i) Máquinas alternativas para llevar a cabo cada operación, de similares características o no. (ii) Tiempo de disponibilidad inicial propio de cada equipo. (iii) Consideración de un tiempo de alistamiento o *changeover* entre pares de operaciones sucesivas de distinto tipo que se ejecutan en cada máquina. (iv) Tiempo de traslado de cada parte entre los equipos. (v) Cuando una operación de manufactura se inicia, no se interrumpe hasta su finalización (*no preemption*).

Asimismo, se realizaron las siguientes suposiciones: (i) Cada *Job* es independiente de los demás y no se consideran prioridades para establecer su orden de procesamiento, ni su tiempo de inicio. (ii) Se dispone de suficientes recursos y dispositivos para el traslado de los *Jobs* entre los equipos. (iii) Se conoce con antelación la duración de cada actividad de mantenimiento preventivo. (iv) Cuando una tarea de mantenimiento preventivo se inicia, la misma no puede ser interrumpida. (v) Las máquinas no sufren rupturas ni desperfectos durante el horizonte de planeación. No se producen fallos en el procesamiento de las partes.

Además, para ubicar las tareas de mantenimiento en el *schedule* se considera que su comienzo se corresponde a una de las siguientes posibles situaciones: (i) El inicio de cada actividad de mantenimiento preventivo se conoce de antemano y se encuentra especificado con precisión. (ii) El comienzo de cada tarea de mantenimiento preventivo ocurre dentro un intervalo establecido mediante cotas. (iii) El fabricante establece las condiciones en las que

recomienda la realización de este tipo de actividades para mantener funcionando cada equipo de forma adecuada.

Así, las tareas de mantenimiento son agendadas en una máquina de acuerdo al nivel de uso de la misma. El mismo puede ser expresado de diferentes formas. Una de éstas considera la acumulación de los tiempos que demandan las operaciones realizadas por el equipo. Así, un equipo requerirá la realización de una tarea de mantenimiento cuando alcance un cierto nivel de uso previsto, expresado como tiempo acumulado de las operaciones que se planifican que el mismo lleve a cabo.

3 DESCRIPCIÓN DEL MODELO

El modelo CP propuesto se ha implementado en el lenguaje OPL de ILOG-IBM, en el entorno de ILOG CPLEX Optimization Studio 12.5 (ILOG–IBM, 2012). Se ha optado por esta tecnología pues facilita la tarea de modelado mediante: (i) variables de tipo intervalo para modelar las tareas, (ii) constructores de alto nivel (e.g., *endBeforeStart(*$task_a$*,*$task_b$*)*, *presenceOf(*$task_a$*)*), (iii) funciones acumulativas para representar la utilización de los recursos y (iv) variables de secuencia para establecer un ordenamiento sobre un conjunto de tareas. Además, posibilita encontrar soluciones factibles iniciales en forma rápida y hallar soluciones óptimas y sub-óptimas en tiempos de CPU reducidos.

4 NOMENCLATURA

4.1 Conjuntos/Índices

J/j,f: partes (*Jobs*) que deben ser procesadas durante el horizonte de planeación.

O/o,q: operaciones que se llevan a cabo en el ambiente FJS.

O_j: conjunto de operaciones demandadas por el *Job j*.

U/u,k: máquinas del FJS.

$U_{o,j}$: subconjunto de máquinas que pueden realizar la operación *o* del *Job j*.

MT/mt: tareas de mantenimiento preventivo.

MT_u: tareas de mantenimiento preventivo requeridas por el equipo *u*.

MR/mr: recursos disponibles para realizar las tareas de mantenimiento preventivo.

4.2 Parámetros

$procTime_{j,o,u}$: tiempo de procesamiento de la operación *o* requerida por el *Job j* en la máquina *u*.

$transpTime_{j,u,k}$: tiempo que insume el traslado del *Job j* entre las máquinas *u* y *k*.

$maintTime_{mt,u}$: duración de la tarea de mantenimiento *mt* en el equipo *u*.

$coTime_{u,o,q}$: tiempo de alistamiento o *changeover* entre dos operaciones sucesivas *o* y *q* que se ejecutan en la misma máquina *u*.

$readyTime_u$: momento a partir del cual se encuentra disponible la máquina *u*.

$relTime_j$: momento a partir del cual puede comenzar la primera operación del *Job j*.

$est_{mt,u}$: tiempo más temprano de inicio de la tarea de mantenimiento *mt* en el equipo *u*.

$lst_{mt,u}$: tiempo más tardío de inicio de la tarea de mantenimiento *mt* en la máquina *u*.

$let_{mt,u}$: tiempo más tardío de fin de la tarea de mantenimiento *mt* en la máquina *u*.

$execOrd_{j,o}$: orden de realización de la operación *o* requerida por el *Job j*.

$initStat_u$: tiempo de uso acumulado de la máquina *u* al inicio del horizonte de planificación, expresado en unidades de tiempo.

$maxUse_u$: máximo nivel de uso acumulado permitido de *u*, sin que ésta reciba una tarea de mantenimiento, expresado en unidades de tiempo.

$minUse_u$: mínimo nivel de uso acumulado de *u*, expresado en unidades de tiempo, a partir del cual *u* puede recibir mantenimiento.

4.3 Variables

$task_{j,o}$: tarea que representa la operación *o* correspondiente al *Job j*.

$machTask_{j,o,u}$: tarea que representa la operación o del *Job j* en la máquina *u*.

$maintTask_{mt,u}$: tarea de mantenimiento *mt* en la máquina *u*.

$resMaintTask_{mt,u,mr}$: tarea de mantenimiento *mt* ejecutada por el recurso *mr* sobre la máquina *u*.

$seqTask_u$: variable de secuencia. Representa el ordenamiento de las tareas asignadas a la máquina *u*.

$transp_j$: variable de secuencia que permite realizar un ordenamiento de las tareas demandadas por el *Job j* considerando las máquinas a las que éstas se asignan.

mk: *Makespan*, tiempo de fin de la última operación agendada en el *schedule*.

tml: *Total Machine load*, carga total en unidades de tiempo de las máquinas del ambiente.

mml: *Maximum machine load*, carga en unidades de tiempo de la máquina más ocupada del ambiente.

Las variables $task_{j,o}$, $machTask_{j,o,u}$, $maintTask_{mt,u}$ y $resMaintTask_{mt,u,mr}$ son de tipo intervalo. Éstas asocian entre sí variables simples que representan el inicio, fin y duración de una tarea. Para acceder a dichos valores se emplean las expresiones $startOf(task_{j,o})$, $endOf(task_{j,o})$ y $sizeOf(task_{j,o})$, respectivamente. Las variables $machTask_{j,o,u}$ y $resMaintTask_{mt,u,mr}$ son variables de intervalo de tipo opcional, por lo que ciertas instancias de las mismas pueden no estar presentes en una solución.

4.4 Funciones acumulativas

$machUse_u$: modela el perfil de uso de cada máquina *u*. En cada equipo sólo una operación de maquinado puede ser llevada a cabo a la vez.

$machMaintUse_u$: representa el uso de una máquina *u* cuando se ejecuta sobre la misma una actividad de mantenimiento.

$resUse_{mr}$: Refleja el perfil de uso del recurso *mr*. Modela la capacidad de un recurso de mantenimiento *mr* de llevar a cabo una única actividad de mantenimiento a la vez.

$machCumulUse_u$: modela el tiempo de uso acumulado de una máquina *u*. Permite representar el tiempo total que la máquina *u* lleva acumulado al realizar tareas de maquinado en el horizonte de planeación actual.

5 MODELO CP

5.1 Restricciones sobre tareas de maquinado

La Expresión (1) asegura que cada tarea de manufactura $task_{j,o}$ se asigne a una única máquina *u*. Para ello, utiliza el constructor *alternative*, el cual fuerza a que sólo una de las instancias de la variable opcional $machTask_{j,o,u}$ participe de la solución.

Es decir, si $machTask_{j,o,u'}$ es igual a 1, entonces $machTask_{j,o,u}$ es igual a 0, $\forall u, u' \in U_{o,j} \wedge u \neq u'$. La instancia de $machTask_{j,o,u}$ presente en la solución se sincroniza con $task_{j,o}$ (tienen iguales tiempos de inicio y fin). El intervalo de $machTask_{j,o,u}$ define entonces la duración de la tarea $task_{j,o}$ cuando se asigna al equipo *u*.

$$alternative\left(task_{j,o}, machTask_{j,o,u}\right) \ \forall j \in J\ , \forall o \in O_j\ , \forall u \in U_{o,j} \qquad (1)$$

Con el fin de restringir a una la cantidad de operaciones que una máquina puede realizar simultáneamente, se plantean las Expresiones (2) y (3). La primera utiliza la función acumulativa $machUse_u$ para modelar el perfil de uso de las máquinas.

En particular, cuando la variable de intervalo $machTask_{j,o,u}$ esté presente en la solución, la función *pulse* hará que $machUse_u$ tome valor uno al inicio del intervalo y vuelva a cero cuando la tarea en cuestión finalice. La restricción (3) evita que se realice más de una operación de maquinado de manera simultánea en la máquina *u*.

$$machUse_u = \sum_{j \in J,\ o \in O_j} pulse\left(machTask_{j,o,u}\ , 1\right) \forall u \in U \tag{2}$$

$$machUse_u \leq 1\ \forall\ u\ \in U \tag{3}$$

La Expresión (4) establece la condición de precedencia entre operaciones consecutivas demandadas por un *Job*. Para ello se emplea el constructor de alto nivel *endBeforeStart*($task_{j,o}$, $task_{j,q}$), equivalente a la sentencia *endOf*($task_{j,o}$) <= *startOf*($task_{j,q}$), el cual indica que la operación *o* debe finalizar antes del inicio de la operación *q*.

$$endBeforeStart\left(task_{j,o}, task_{j,\ q}\right) \tag{4}$$

$$\forall\ j \in J\ ,\ \forall o, q \in O_j,\ execOrd_{j,o} + 1 = execOrd_{j,q}$$

Una tarea de maquinado $task_{j,o}$ podrá ejecutarse luego del tiempo de disponibilidad inicial *ready-time* del equipo asignado a la misma. La Restricción (5) modela esta situación, empleando la función *presenceOf*($machTask_{j,o,u}$), que toma el valor 1 cuando la variable está presente en la solución y 0 en caso contrario.

$$startOf\left(task_{j,o}\right) \geq readyTime_u * presenceOf\left(machTask_{j,o,u}\right) \tag{5}$$

$$\forall\ j \in J,\ \forall o \in O_j,\ \forall u \in U_{o,j}$$

La restricción (6) establece que toda tarea $task_{j,o}$ comience luego del tiempo de liberación del *Job j* (*release time*).

$$startOf\left(task_{j,o}\right) \geq relTime_j \quad \forall\ j\ \in J, \quad \forall\ o\ \in O_j \tag{6}$$

La Expresión (7) utiliza el constructor de alto nivel *noOverlap*($seqTask_u$, $coTime_{u,o,q}$) que (i) evita el solapamiento de las tareas asociadas a la variable de secuencia, y (ii) fuerza un tiempo de *changeover* $coTime_{u,o,q}$ entre las operaciones *o* y *q*.

$$noOverlap\left(seqTask_u, coTime_{u,o,q}\right) \quad \forall\ u\ \in U, \quad \forall\ o, q\ \in O \tag{7}$$

Cada *Job j* requiere ser trasladado entre las máquinas *u* y *k* que ejecutan las operaciones consecutivas *o* y *q*. La Expresión (8), mediante el empleo del constructor de alto nivel *noOverlap*, fuerza a que exista un tiempo de transporte del *Job j* entre la finalización de la operación *o* en la máquina *u* y el inicio de *q* en el equipo *k*.

$$noOverlap\left(transp_j, transpTime_{j,u,k}\right) \quad \forall j \in J, \quad \forall u,k \in U, \quad u \neq k \tag{8}$$

5.2 Restricciones sobre tareas de mantenimiento preventivo

La Restricción (9) asegura que una tarea de mantenimiento preventivo *mt* que se realiza en un equipo *u* utilice sólo un recurso o cuadrilla de mantenimiento *mr*. Al igual que en la Expresión (1) se emplea la expresión *alternative*, que vincula las variables $resMaintTask_{mt,u,mr}$ y $maintTask_{mt,u}$.

$$alternative\left(maintTask_{mt,u}, resMaintTask_{mt,u,mr}\right) \quad \forall mt \in MT_u, \quad \forall mr \in MR, \quad \forall u \in U \tag{9}$$

Las cuadrillas de mantenimiento *mr* son recursos unarios. La Expresión (10) representa el perfil de uso de los recursos *mr*, mediante la utilización de la función acumulativa $resUse_{mr}$ y el predicado *pulse*. La Expresión (11) impide que cada cuadrilla realice más de una actividad en simultáneo.

$$resUse_{mr} = \sum_{mt \in MT_u,\ u \in U} pulse\left(resMaintTask_{mt,u,mr}, 1\right) \quad \forall mr \in MR \tag{10}$$

$$resUse_{mr} \leq 1 \quad \forall mr \in MR \tag{11}$$

Cuando las tareas de mantenimiento $maintTask_{mt,u}$ comienzan en un momento específico establecido de antemano, o en algún instante perteneciente a un intervalo previamente definido por cotas, las Restricciones (12) y (13) modelan estas situaciones. Cuando la actividad de mantenimiento debe iniciarse en un instante predefinido, el valor de $est_{mt,u}$ y de $lst_{mt,u}$ coinciden.

$$startOf\left(maintTask_{mt,u}\right) \geq est_{mt,u} \quad \forall mt \in MT_u, \forall u \in U \tag{12}$$

$$startOf\left(maintTask_{mt,u}\right) \leq lst_{mt,u} \quad \forall mt \in MT_u, \forall u \in U \tag{13}$$

En caso de que las tareas de mantenimiento preventivo *mt* se ejecuten de acuerdo al nivel de empleo planificado de cada máquina, se utiliza la función acumulativa $machCumulUse_u$ definida en la Expresión (14).

Como en los problemas de *scheduling* se trabaja con horizontes rodantes, puede ocurrir que los equipos posean un tiempo de uso acumulado al inicio del presente período de planificación.

En el primer término de la Expresión, mediante el uso de la función *step*, se toma el valor del parámetro $initialStatus_u$ como el valor inicial de la función $machCumulUse_u$. El segundo término indica que cuando la variable $machTask_{j,o,u}$ se encuentra presente en la solución, la función *stepAtStart* actualiza la función acumulativa incrementando su valor en una cantidad igual al tiempo de dicha tarea.

El tercer término se emplea para establecer que sólo puede realizarse una tarea de mantenimiento preventivo cuando el nivel de uso de la máquina *u* es superior a $minUse_u$ e inferior a $maxUse_u$.

$$machCumulUse_u = step\left(0, initialStatus_u\right) + \sum_{j \in J,\ o \in O} stepAtStart\left(machTask_{j,o,u}\ ,\ procTime_{j,o,u}\right) - \sum_{mt \in MT} stepAtStart\left(maintTask_{mt,u}\ ,\ minUse_u,\ maxUse_u\right) \quad (14)$$

$$\forall\, u\ \in U$$

Al finalizar una tarea de mantenimiento preventivo en una máquina *u*, el tiempo de uso acumulado de la misma, capturado por la función $machCumulUse_u$, vuelve a cero, comenzando desde ese momento a acumular nuevamente su tiempo de uso. Esto queda garantizado mediante la Expresión (15) que utiliza la función *alwaysIn*.

Así, *alwaysIn(cumulFunction,intervalTask,x,y)* establece que si la instancia de la variable *intervalTask* se encuentra presente en la solución, la función *cumulFunction* adopta un valor del rango definido por [*x*,*y*] entre el inicio y final de *intervalTask*.

$$alwaysIn\left(machCumulUse_u, maintTask_{mt,u}, 0, 0\right) \quad (15)$$

$$\forall u \in U, \forall mt \in MT_u$$

La restricción (16) fuerza que el tiempo de uso acumulado de cada máquina en ningún momento pueda superar el valor máximo $maxUse_u$ establecido de antemano.

$$machCumulUse_u \leq\ maxUse_u\ \ \forall\, u\ \in U \quad (16)$$

Las Restricciones (17) y (18) se utilizan para impedir que las tareas de procesamiento y las actividades de mantenimiento se lleven a cabo en forma simultánea en un mismo equipo. Para ello se utiliza la función acumulativa $machMaintUse_u$ definida en la Expresión (17). La Restricción (18) evita el solapamiento de ambos tipos de actividades que hacen uso de la capacidad unaria del equipo.

$$machMaintUse_u = \sum_{mt \in MT_u} pulse\left(maintTask_{mt,u}, 1\right) \forall u \in U \tag{17}$$

$$machUse_u + machMaintUse_u \leq 1 \ \forall u \in U \tag{18}$$

5.3 Funciones objetivo

Se han contemplado como medidas de desempeño a optimizar la minimización del Makespan (19), Carga total de los equipos (20) y Carga máxima del equipo con mayor ocupación en el ambiente de manufactura (21). Según el criterio que se quiera optimizar, una de las siguientes expresiones debe ser agregada al modelo.

$$mk = \max\left(endOf\left(task_{j,o}\right)\right) \tag{19}$$

$$\forall\, j \in J\ , \forall o \in O$$

$$tml = \sum_{\forall\, j \in J\ ,\ o \in O} sizeOf\left(task_{j,o}\right) \tag{20}$$

$$mml = \max\left(\sum_{\forall\, j \in J\ ,\ o \in O} sizeOf\left(machTask_{j,o,u}\right)\right) \tag{21}$$

$$\forall u \in U$$

5.4 Función multiobjetivo

Cuando se consideran medidas de desempeño conflictivas entre sí, resulta de interés el planteo de una función multiobjetivo (Expresión 22) que permita hallar soluciones no dominadas o soluciones de Pareto. La función fue normalizada y se compone de tres términos asociados a los criterios de desempeño evaluados.

Cada uno de ellos se ha afectado con un factor de peso p_i que pondera la medida de desempeño i. Los factores de peso cumplen la Restricción (23).

$$mo = p_1 * \frac{\left(mk_{max} - mk\right)}{mk_{max} - mk_{min}} + p_2 * \frac{\left(tml_{max} - tml\right)}{tml_{max} - tml_{min}} + p_3 * \frac{\left(mml_{max} - mml\right)}{mml_{max} - mml_{min}} \tag{22}$$

$$\sum_{i \in Criterios\ evaluados} p_i = 1 \tag{23}$$

El valor máximo de cada criterio de desempeño corresponde al peor valor obtenido cuando se minimiza alguna de las otras medidas de desempeño. El valor mínimo de cada uno corresponde al valor hallado al optimizar únicamente ese criterio de desempeño.

6 RESULTADOS COMPUTACIONALES

El modelo CP propuesto ha sido validado mediante diversos problemas FJS que pueden encontrarse en la bibliografía del área. Los ejemplos abordados se diferencian entre sí por la cantidad de *Jobs* (entre 4 y 20), las operaciones requeridas por cada *Job* (entre 3 y 14), equipos disponibles (entre 4 y 15), tipo de FJS (Total FJS o Parcial FJS), las tareas de mantenimiento preventivo (1 o 2 tareas por equipo), los recursos disponibles para llevarlas a cabo, los criterios de inicio de las tareas, entre otras características.

Los casos de estudio presentados en la Tabla 1 han sido seleccionados entre un número mayor de ejemplos resueltos, para demostrar que el modelo CP permite obtener soluciones de buena calidad en diversas situaciones.

Tabla 1 – Descripción de casos de estudio abordados

Denominación de Caso	Jobs	Oper. por Job	Máquinas	Categoría	Mant.	Tareas de mant. por equipo	Recursos de mant.
C1 – I1	4	2 – 4	5	T-FJS	NO	-	-
C1 – I2	10	2 – 3	7	T-FJS	NO	-	-
C1 – I3	10	3	10	T-FJS	NO	-	-
C1 – I4	15	2 – 4	10	T-FJS	NO	-	-
C2	15	5	5	T-FJS	SÍ	Variables	2
C3	8	3 – 4	8	P-FJS	SÍ	1	3

T-FJS. Cada operación de cada *Job* puede realizarse en todos los equipos
P-FJS. Al menos una operación de un *Job* sólo puede realizarse en un subconjunto de equipos

Caso de estudio C1: corresponde a los problemas I1, I2, I3 e I4 descriptos en el trabajo de Kacem, Hammadi y Borne (2002). El ambiente posee entre 5 y 10 equipos, dependiendo el caso abordado. Las operaciones pueden realizarse en equipos alternativos disímiles. Cada *Job* posee un *release-time*. No se considera la existencia de tareas de mantenimiento preventivo planificado. Las medidas de desempeño utilizadas son *makespan*, carga de equipos y carga máxima de equipos del ambiente. También se emplea una función multiobjetivo que considera la optimización de los tres criterios anteriores en simultáneo. Los datos completos de estos ejemplos pueden encontrarse en Repositorio de datos (2014).

Caso de estudio C2: Corresponde a una modificación del denominado *MPM job-shop scheduling problem with maintenance activities* presentado en Wang y Yu (2010). Se trata de

un ambiente con 5 máquinas multipropósito que ejecutan las operaciones demandadas por 15 *Jobs*. Cada *Job* requiere de 5 operaciones, las que cuentan con equipos alternativos y disímiles para su realización. Se consideran tareas de mantenimiento preventivo, las que se ubican en la agenda de trabajo de acuerdo al nivel de uso acumulado que se planifica para cada equipo. Se dispone de dos cuadrillas para realizar tareas de mantenimiento. La medida de desempeño considerada es *makespan*. Los datos correspondientes se encuentran disponibles en el Repositorio de datos (2014).

Caso de estudio C3: modificación del ejemplo denominado *Problem 8x8/27 operations with maintenance activities* descripto en Wang y Yu (2010). Éste consiste en un conjunto de 8 *Jobs*, con 3 o 4 operaciones cada uno, las que pueden llevarse a cabo en 8 máquinas. En cada una se realiza una única tarea de mantenimiento preventivo cuyo inicio se encuentra especificado por cotas. Se cuenta con recursos para efectuar hasta tres tareas de mantenimiento en simultáneo. El criterio de desempeño a optimizar es *makespan*. En este ejemplo se consideran la mayor parte de las características de los ambientes FJS; entre éstas, la existencia de máquinas alternativas disímiles para llevar a cabo las operaciones, *ready-times* de equipos, tiempos de alistamiento dependientes de la secuencia de tareas, tiempos de traslado de *Jobs* entre máquinas. Los datos correspondientes pueden encontrarse en Repositorio de datos (2014).

Los resultados reportados en la Tabla 2 corresponden a la ejecución del modelo en una PC con 8 GB de RAM, procesador *Intel(R) Xeon(R)* 3,10 GHz, utilizando la estrategia de búsqueda por defecto que provee el resolvedor de IBM-ILOG. Se estableció el límite de 900 s de CPU para encontrar soluciones para cada caso.

Tabla 2 – Resultados computacionales para un conjunto de los ejemplos considerados

Problema	**Función Objetivo**					
	Makespan		**Carga Total**		**Carga Máxima**	
	FO	**t [s]**	**FO**	**t [s]**	**FO**	**t [s]**
C1 – I1	16	0,10	32	0,14	7	0,46
C1 – I2	15	0,76	60	1,13	10[a]	9,50
C1 – I3	7	0,36	41	1,62	5[a]	58,98
C1 – I4	23	1,27	91	5,13	10	23,21
C2	702[a]	879,8	3406[a]	211,1	722[a]	317,84
C3	174[a]	20,59	730	0,59	110[a]	10,58

[a]: solución subóptima no comprobada.
FO: Valor de la Función Objetivo

Cuando se resolvió el caso C1 se encontraron los mismos resultados que los reportados en Kacem, Hammadi y Borne (2002) para todas las instancias y las diferentes funciones objetivo consideradas, excepto para un único ejemplo (I2-Carga máxima), donde el valor obtenido fue una unidad mayor. En la mayor parte de los ejemplos se arribó a soluciones óptimas o subóptimas en tiempos de CPU bajos (en el trabajo de Kacem, Hammadi y Borne (2002) éstos no se informaron).

En el ejemplo C2, al minimizar *makespan*, se alcanzó una solución subóptima con un *makespan* de 702 unidades de tiempo (u.t.) en 879,8 s. La solución indica que se requieren 5 tareas de mantenimiento en total y les asigna un tiempo de inicio y duración a cada una dependiente de la máquina. Es interesante destacar que la primera solución hallada por el modelo, de 828 u.t., se instanció de inmediato en sólo 0,39 s.

En el ejemplo C3 se obtuvo una solución subóptima con un *makespan* de 174 (u.t) a los 20,59 s y no fue posible mejorar la misma en el tiempo límite de 900 s. El modelo permitió alcanzar una primera solución de 245 (u.t) en apenas 0,27 s y logró ubicar las tareas de procesamiento y mantenimiento según las especificaciones del problema, de manera eficiente. Por razones de espacio, no se publican aquí los diagramas de Gantt ilustrativos de cada solución.

En la Tabla 3 se reportan todas las soluciones no dominadas (Ver definición en Repositorio de datos (2014)) obtenidas con la formulación propuesta para los casos C1 al adoptar la función (22). Se variaron los factores de peso *pi* de cada criterio para obtener diferentes soluciones para cada ejemplo.

Tabla 3 – Resultados obtenidos por el modelo CP al optimizar la función multiobjetivo propuesta y valores reportados en Kacem, Hammadi y Borne (2002)

Problema	**Modelo CP**				**Kacem y colab.**			
		mk	**tml**	**mml**		**mk**	**tml**	**mml**
C1 – I1	S_{I1-1}	16	33	7	S_{I1-A}	18	33	7
	S_{I1-2}	16	32	8	S_{I1-B}	18	32	8
					S_{I1-C}	16	35	9
					S_{I1-D}	16	34	10
C1 – I2	S_{I2-1}	15	61	11	S_{I2-A}	15	61	11
	S_{I2-2}	15	62	10	S_{I2-B}	17	64	10
	S_{I2-3}	16	60	12	S_{I2-C}	18	63	10
					S_{I2-D}	16	66	10
					S_{I2-E}	16	60	12
C1 – I3	S_{I3-1}	7	43	5	S_{I3-A}	7	45	5
	S_{I3-2}	8	42	5	S_{I3-B}	8	42	5
	S_{I3-3}	8	41	7	S_{I3-C}	8	41	7
	S_{I3-4}	7	42	6				
C1 – I4	S_{I4-1}	23	93	10	S_{I4-A}	23	95	11

Problema	Modelo CP				Kacem y colab.			
	S_{I4-2}	23	91	11	S_{I4-B}	24	91	11

Asimismo, se reportan los resultados hallados por Kacem y colab. (KACEM; HAMMADI; BORNE, 2002) para el mismo caso de estudio. Debe remarcarse que 10 de las 14 soluciones reportadas son de baja calidad, ya que son dominadas. Específicamente, se advierte que en el caso C1-I1 la solución S_{I1-1} encontrada por el modelo CP domina a S_{I1-A} y de manera similar, S_{I1-2} domina a S_{I1-B}, S_{I1-C} y S_{I1-D}.

En el ejemplo C1-I2 S_{I2-2} domina a las soluciones S_{I2-B}, S_{I2-C} y S_{I3-D}. Asimismo, en el caso C1-I3 se puede advertir que S_{I3-1} domina a S_{I3-A}. Finalmente, en el ejemplo C1-I4 se aprecia que las soluciones halladas con el modelo CP propuesto, S_{I4-1} y S_{I4-2}, dominan a S_{I4-A} y S_{I4-B}, respectivamente. Por el contrario, las 11 soluciones halladas con el modelo CP son no dominadas.

7 CONCLUSIONES

Se ha presentado una formulación basada en CP para abordar el problema de *scheduling* predictivo en ambientes FJS donde se deben programar conjuntamente tareas de procesamiento y actividades de mantenimiento preventivo. Además de las características propias de los ambientes FJS, se consideraron diversos escenarios referidos al posible comienzo de las tareas de mantenimiento que se demanden: inicio fijo, inicio variable acotado e inicio sujeto al uso acumulado de los equipos. La propuesta hace uso de las ventajas del enfoque CP para problemas de *scheduling*, permitiendo contar con un modelo extensible y adaptable fácilmente a diferentes configuraciones de ambientes FJS.

El modelo se validó con numerosos ejemplos provenientes de la literatura, así como casos de estudio creados a partir de los mismos. Se emplearon las medidas de desempeño *makespan*, carga total de máquinas y carga máxima de máquinas, así como una función multiobjetivo que las incluye. Se hallaron soluciones óptimas y subóptimas en bajos tiempos de CPU, encontrándose soluciones de igual o mejor calidad que las reportadas en la bibliografía.

Las agendas de trabajo obtenidas muestran los beneficios de efectuar conjuntamente la programación de las tareas de manufactura y de mantenimiento preventivo. En el futuro se evaluará el modelo de manera más exhaustiva estudiando, por ejemplo, su escalabilidad. Además, se planea abordar el problema de *scheduling* dinámico empleando como base la presente formulación.

SHORT-TERM PRODUCTION PLANNING WITH PREVENTIVE MAINTENANCE TASKS CONSIDERATION IN FLEXIBLE JOB SHOP ENVIRONMENTS

ABSTRACT: This work deals with the predictive scheduling problem in industrial plants of Flexible Job Shop type. To deal with it a Constraint Programming (CP) model has been developed. It allows obtaining an efficient agenda for a set of jobs that is known beforehand. The model takes into account the characteristic features of this industrial environment: dissimilar manufacturing routes for each job, multipurpose machines, changeover times, machine ready times, etc. In addition, the proposal considers the need to perform predictive maintenance tasks in each unit. The model has been verified and validated by means of medium size examples available in the literature. Solutions of very good quality have been found in reduced CPU times. These findings allow to conclude that the model is very competitive.

Keywords: Predictive scheduling. Flexible job shop. Constraint programming.

REFERENCIAS

BRAILSFORD, S.; POTTS, C.; SMITH, B. Constraint satisfaction problems: Algorithms and applications. **European Journal of Operational Research**, v. 119, p. 557-581, 1999.

GAO, J.; SUN, L.; GEN, M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. **Computers & Operations Research**, v. 35, p. 2892-2907, 2008.

GAREY, M.R.; JOHNSON, D.S.; SETHI, R. The Complexity of Flow-shop and Job-shop Scheduling. **Mathematics of Operations Research**, v. 1, p. 117-129, 1976.

ILOG–IBM. **IBM ILOG CPLEX Optimization Studio**. 2012. Disponible en: <http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/>. Accedido: 31/03/2014.

KACEM, I.; HAMMADI, S.; BORNE, P. Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems. **IEEE Transactions on Systems, Man and Cybernetics**, v. 32, p. 1-13, 2002.

KARIMI, H.; RAHMATI, S.H; ZANDIEH, M. An efficient knowledge-based algorithm for the flexible job shop scheduling problem. **Knowledge-Based Systems**, v. 3, p. 236-244, 2012.

LEUNG, J.Y-T. **Handbook of scheduling:** algorithms, models and performance analysis. Chapman-Hall/CRC, 2004.

REPOSITORIO DE DATOS. Disponible en: <https://sites.google.com/site/43jaiiosnh/>. Accedido: 23/04/2014.

WANG, S.; YU, J. An effective heuristic for flexible job-shop scheduling problem with maintenance constraints. **Computers and Industrial Engineering**, v. 59, p. 436-447, 2010.

XIE, Z., HAO, S., YE, G., TAN, G. A new algorithm for complex product flexible scheduling with constraint between Jobs. **Computers and Industrial Engineering**, v. 57, p. 766-772, 2009.

Originais recebidos em: 23/04/2015

Aceito para publicação em: 02/05/2016