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

PLANIFICACIÓN DE LA PRODUCCIÓN DE CORTO PLAZO EN PLANTAS "*BATCH*" MULTIPRODUCTO MULTIETAPA: UN ENFOQUE CP NOVEDOSO

Franco Matías Novara[1]

Juan Novas[2]

Gabriela Henning[3]

RESUMEN: Este trabajo se ocupa del problema de "*scheduling*" predictivo de una planta industrial "*batch*" multiproducto, multietapa con unidades disímiles operando en paralelo en cada etapa. Se presenta un modelo de programación con restricciones (CP) que posibilita abordar de manera eficiente este problema. El mismo permite modelar una amplia variedad de características y limitaciones que están presentes en este tipo de ambiente industrial. Se reportan los resultados correspondientes a varios ejemplos de tamaño mediano que ya han sido abordados en la bibliografía del área. Éstos demuestran que el desempeño computacional del modelo es muy bueno, pues se encuentran soluciones óptimas/subóptimas de muy buena calidad en bajos tiempos de CPU.

Palabras claves: *Scheduling* predictivo. Plantas *batch*. Programación con restricciones. Recursos limitados.

1 INTRODUCCIÓN

En diversas ramas industriales (química fina, alimenticia, farmacéutica, etc.) se emplean procesos de fabricación *batch* multiproducto, multietapa. En estos ambientes se manufacturan productos de una misma familia, que requieren iguales etapas de procesamiento, mediante una política de producción por lotes o *batches*. Así, cuando se fabrica un *batch* de un producto,

[1] Estudiante de Ingeniería Industrial, Facultad de Ingeniería Química, UNL, Santiago del Estero 2829, 3000 Santa Fe, Argentina, Instituto de Desarrollo Tecnológico para la Industria Química (INTEC), Universidad Nacional del Litoral, Consejo Nacional de Investigaciones Científicas y Técnicas (UNL, CONICET), Güemes 3450, 3000 Santa Fe, Argentina E-mail: fra_novara@hotmail.com.

[2] Dr. Instituto de Desarrollo Tecnológico para la Industria Química (INTE), Universidad Nacional del Litoral, Consejo Nacional de Investigaciones Científicas y Técnicas (UNL, CONICET), Güemes 3450, 3000 Santa Fe, Argentina. Dirección actual: CIEM-CONICET/UTN-FRC, Ciudad Universitaria, X5000HUA Córdoba, Argentina. E-mail: mnovas@intec.unl.edu.ar .

[3] Dra. Instituto de Desarrollo Tecnológico para la Industria Química (INTEC), Universidad Nacional del Litoral, Consejo Nacional de Investigaciones Científicas y Técnicas (UNL, CONICET), Güemes 3450, 3000, Santa Fe, Argentina. E-mail: ghenning@intec.unl.edu.ar.

éste atraviesa todas las etapas del proceso de manera secuencial, realizándose una tarea de procesamiento en cada etapa. Generalmente se dispone de unidades que operan en paralelo en las etapas, dando lugar a ambientes denominados *job-shop* flexibles (MARAVELIAS, 2012).

Para generar la agenda de trabajo de los mismos es necesario: (i) asignar un equipo a cada tarea del *batch*, (ii) secuenciar las tareas asignadas a una misma unidad y (iii) establecer sus tiempos de inicio y fin, de manera de optimizar una medida de desempeño. La obtención de esta agenda constituye el problema abordado en este trabajo. Éste cobra importancia debido a la multiplicidad de industrias que operan con este tipo de proceso productivo y a la dificultad intrínseca del mismo, ya que se trata de un problema combinatorio de tipo NP-*hard* (*nondeterministic polynomial-bounded – hard*).

A ello se suma la variedad de aspectos que deben ser contemplados, tales como unidades de procesamiento disímiles en las etapas; restricciones topológicas, requerimiento de otros recursos discretos (i.e. aquéllos cuya capacidad se expresa como un número entero positivo) disponibles en cantidades limitadas, fechas de entrega de las órdenes (*due-dates*), fechas de disponibilidad de órdenes (*release-times*) y equipos (*ready-times*), tiempos de alistamiento de equipos y de limpieza dependientes de la secuencia (*changeover-times*), distintas políticas de almacenamiento intermedio/operación entre etapas, etc.

Este problema ha despertado un gran interés en la comunidad científica, la cual ha desarrollado principalmente enfoques basados en programación lineal mixta entera (MILP) – (CASTRO; GROSSMANN, 2005) y (MARCHETTI; CERDÁ, 2009) –, y, en menor medida, programación con restricciones (*Constraint Programming*, CP) (ZEBALLOS; NOVAS; HENNING, 2011). También existen aportes que combinan ambas propuestas y potencian sus ventajas (JAIN; GROSSMANN, 2001; HARJUNKOSKI; GROSSMANN, 2002). Una revisión reciente de los principales enfoques puede encontrarse en Méndez et al. (2006).

El modelo presentado en este trabajo se basa en un enfoque CP con capacidad de contemplar los aspectos antes citados y también limitaciones en la disponibilidad de equipos de limpieza y de ciertos recursos renovables (electricidad, vapor, mano de obra, etc.), requeridos por los *batches*. Además, permite adoptar diferentes funciones objetivo, tanto las vinculadas al tiempo de finalización de las tareas, como otras relacionadas a los costos de operación de la planta, de manera eficiente. Las razones por las cuales se ha optado por desarrollar un modelo CP en lugar de uno de tipo MILP se han discutido en detalle en Zeballos, Novas y Henning (2011).

En la Sección 2 de este trabajo se describe en detalle el problema abordado, así como las suposiciones realizadas. Luego se presenta el modelo elaborado que fue dividido en tres partes: un modelo básico y dos extensiones. En la Sección 4 se discuten resultados computacionales, comparándolos con los alcanzados por otros autores. Finalmente, se presentan conclusiones y trabajos futuros.

2 DESCRIPCIÓN DEL PROBLEMA

Se aborda el problema de programación de la producción de corto plazo (*scheduling* predictivo) de una planta *batch* multiproducto, multietapa, en la que existen equipos que operan en paralelo en cada etapa, los que pueden tener características diferentes (tiempos de procesamiento distintos para una misma tarea). En este ambiente, las unidades de procesamiento de una etapa no necesariamente están conectadas con todas las unidades de las etapas previa y posterior, por lo cual es necesario considerar la topología de la planta. Asimismo, en ciertas etapas del proceso se requieren otros recursos adicionales (mano de obra, electricidad, vapor), los cuales están disponibles en cantidades limitadas; estos recursos se consideran recursos discretos renovables.

Al inicio del horizonte de planificación se cuenta con información de las órdenes a incluir en la agenda, cada una de las cuales corresponde a un *batch* o lote de un producto diferente, que se procesará secuencialmente en la planta, no permitiéndose división y/o mezclado de lotes. El problema consiste en asignar, secuenciar y crear la agenda de las tareas que requiere cada *batch*; es decir, definir qué equipo procesará cada una de ellas, secuenciar las tareas asignadas a una misma unidad y establecer los tiempos de inicio y fin de cada actividad.

Al resolver este problema debe considerarse que al comienzo del horizonte temporal pueden haber equipos que aún no están listos para operar y que puede haber *batches* que aún no pueden procesarse. Cada equipo requiere un tiempo de alistamiento o *set-up*, independiente del producto que procese y un tiempo de limpieza o *changeover* que es función de la secuencia de productos (del que procesó anteriormente y del que se va a procesar).

Además, para realizar este *changeover* se requiere un equipo de limpieza del cual se dispone un número limitado de unidades. También es posible que existan secuencias de procesamiento de *batches* prohibidas en ciertos equipos. Finalmente, deben considerarse las políticas de almacenamiento intermedio/operación entre etapas. En este trabajo se contempla (i) trabajar sin restricciones de almacenamiento intermedio, suponiendo que existe capacidad

ilimitada (UIS – *Unlimited Intermediate Storage*), así como la situación opuesta, considerar que no existen tanques de almacenamiento intermedio (NIS – *Non-Intermediate Storage*), (ii) permitir que un *batch* espere en la unidad que lo procesó hasta que la siguiente unidad esté disponible (UW – *Unlimited Wait*), dando lugar a una política NIS/UW, o (iii) no permitir esperas (ZW – *Zero-Wait*), dando lugar a una política NIS/ZW. Las suposiciones adicionales realizadas al elaborar el modelo son:

- La transferencia de cada *batch* entre etapas es instantánea.
- La disponibilidad de los recursos discretos es constante a lo largo del horizonte temporal.
- Los requerimientos de recursos renovables como mano de obra, electricidad, etc., son función del *batch* y de la etapa, siendo independientes del equipo.

3 DESCRIPCIÓN DEL MODELO

El modelo CP presentado a continuación fue implementado en el lenguaje OPL soportado por el paquete IBM ILOG CP Optimizer y el entorno de desarrollo IBM ILOG CPLEX *Optimization Studio* 12.4 (IBM, 2011). Se ha optado por este ambiente debido a las facilidades que ofrece para abordar problemas de *scheduling*, entre las que se destacan: (i) inclusión de variables y constructores de alto nivel que simplifican las restricciones, (ii) elevada flexibilidad en el modelado y expresividad del lenguaje, generada a partir de la posibilidad de utilizar expresiones algebraicas lineales y no lineales, lógicas y mixtas; (iii) rapidez para obtener buenas soluciones iniciales, así como para detectar situaciones donde no hay una solución factible, (iv) capacidad para manejar eficientemente gran cantidad de variables; y (v) posibilidad de definir estrategias de búsqueda basadas en el conocimiento del dominio de trabajo, que incrementan, en ciertos casos, la velocidad de instanciación de buenas soluciones y/o de hallazgo de una solución óptima.

Dentro de los constructores de alto nivel, que el lenguaje ofrece y que fueron utilizados en este trabajo, se encuentran:

- **Variable de intervalo**: representa un intervalo de tiempo en el cual ocurre una actividad. Se caracteriza por sus atributos de inicio, fin y tamaño, a cuyos valores puede accederse utilizando las funciones *startOf*, *endOf* y *sizeOf*. Además, puede declararse como opcional. En tal caso puede estar o no presente en la solución final;

la función *presenceOf* devuelve 1 cuando la variable está presente en la solución final y cero en caso contrario.

- **Variable de secuencia**: esta variable genera un ordenamiento lógico de un conjunto de variables de intervalo. El constructor *noOverlap* asegura que ese ordenamiento lógico coincida con el orden temporal de las variables; y además permite modelar el tiempo de *changeover*, que debe separar el tiempo de finalización de una tarea con el inicio de la siguiente en la secuencia, en un dado equipo.
- ***typeOfNext***: dada una variable de secuencia S y otra de intervalo V, esta variable devuelve el tipo asociado a la variable de intervalo que está a continuación de V en la secuencia S. El tipo de una variable corresponde al valor de un atributo que la caracteriza, y que ha sido previamente asociado a la misma. Un ejemplo sencillo se refiere al tipo de producto que se asocia a un *batch.*
- ***cumulFunction*** permite representar la utilización acumulada de los recursos limitados, demandados por parte de las distintas actividades, en función del tiempo. Existen dos tipos: **step** y **pulse**; según sea que el incremento en su valor se produce con el inicio/final de una tarea y permanezca en el tempo (*step*), o sólo ocurra durante la ejecución de la misma y luego disminuya (*pulse*).
- ***alwaysIn*** permite establecer el valor que tiene que tomar una determinada función acumulativa durante la ejecución de una tarea que es representada por una variable de intervalo.
- ***endBeforeStart*** y ***endAtStart*** son constructores que permiten establecer precedencias para secuenciar tareas (variables de intervalo) en el tiempo.

3.1 Nomenclatura

A continuación se presenta la nomenclatura utilizada: conjuntos e índices, parámetros, funciones acumulativas y variables. En el caso de las dos últimas, oportunamente se indica el tipo de función acumulativa (*step* o *pulse*) y el tipo de variable (de intervalo, secuencia o entera).

3.1.1 Conjuntos/índices

Batches/b: *batches* a procesar en el período.

Stages/s: etapas del proceso productivo.

Units/u: unidades de procesamiento.

$Units_s$: unidades de procesamiento que pertenecen a la etapa *s*.

F_u: conjunto de unidades de la etapa *s+1* no conectadas con la unidad *u* de la etapa *s*.

Resources/r: recurso productivo discreto (electricidad, vapor, etc.).

Fb: conjunto de pares de *batches* que implican transiciones prohibidas, $f=<b_1,b_2>$.

3.1.2 Parámetros

$pt_{b,u}$: tiempo de procesamiento del *batch b* en el equipo *u*.

$co_{b1,b2}$: tiempo de *changeover* entre el *batch* b_1 y el b_2.

su_u: tiempo de alistamiento o *setup* de la unidad *u*.

rd_u: fecha a la cual está disponible la unidad *u*; *ready-time* de *u*.

rt_b: *release-time* del *batch b*.

$avail_r$: cantidad disponible del recurso *r* durante todo el horizonte de planificación.

$requir_{b,s,r}$: requerimiento del recurso *r* en la etapa *s* por parte del *batch b*.

3.1.3 Funciones acumulativas

$unitUsage_u$: función acumulativa tipo *pulse* que modela la capacidad de los equipos de procesar como máximo un *batch* o lote por vez.

$forbiddenChangeOver_{b,u}$: función acumulativa tipo *step* que se define a efectos de modelar secuencias prohibidas.

$resourceUsage_r$: función acumulativa tipo *pulse* que modela la disponibilidad limitada del recurso discreto *r*.

cleaningResourceUsage: función acumulativa tipo *pulse* que modela la disponibilidad de los recursos de limpieza.

3.1.4 Variables

$task_{b,u}$: tarea de procesamiento del *batch b* en la unidad *u* (variable de intervalo).

$cleanTask_{b1,b2,u}$: tarea de limpieza, llevada a cabo en la unidad *u* al finalizar la tarea de procesamiento del *batch* b_1, antes de iniciar el procesamiento de b_2 (variable de intervalo).

$taskSequence_u$: variable de secuencia asociada a las tareas del equipo *u*.

Mk: *Makespan* de la agenda de trabajo (variable entera).

3.2 Modelo básico

Las variables $task_{b,u}$ y $cleanTask_{b1,b2,u}$ son variables de intervalo que representan los intervalos de tiempo durante los cuales se desarrollan las tareas de procesamiento y limpieza, respectivamente. Ambas se asocian a tres variables simples, dos de ellas independientes, que representan el inicio, final y duración del intervalo. Para acceder a cada una de ellas se utilizan las sentencias *startOf($task_{b,u}$)*, que retorna el inicio del intervalo), *endOf($task_{b,u}$)*, que retorna el fin y *sizeOf($task_{b,u}$)*, que representa la duración.

Además, se utiliza el predicado *presenceOf($task_{b,u}$)*, el cual retorna valor 1 ó 0 en correspondencia con la presencia o no de la variable. Este predicado se emplea en la Expresión 1 para asegurar que cada tarea se asigne a un único equipo por etapa y en la expresión 2 para establecer la duración de la tarea. En caso que entre las etapas se adopte una política NIS/UW es necesario cambiar el signo de la Expresión 2 por ≥.

$$\sum_{u \in Units_s} presenceOf\left(task_{b,u}\right) = 1\,, \forall\, s \in Stages, \forall\, b \in Batches \tag{1}$$

$$sizeOf\left(task_{b,u}\right) = pt_{b,u} \cdot presenceOf\left(task_{b,u}\right), \forall\, b \in Batches, \forall\, u \in Units \tag{2}$$

Para establecer la condición de precedencia entre las tareas que pertenecen a un mismo *batch* se plantea la restricción (Expresión 3). En ella se utiliza el constructor *endBeforeStart*, el cual es equivalente a *endOf($task_{b,u1}$) ≤ startOf($task_{b,u2}$)* cuando ambas variables están presentes (sus predicados *presenceOf* asociados valen 1). En caso de adoptarse una política de almacenamiento intermedio/operación NIS/ZW o NIS/UW entre etapas es necesario cambiar el constructor *endBeforeStart* por *endAtStart*; que equivale a la Expresión *endOf($task_{b,u1}$) = startOf($task_{b,u2}$)*.

$$\begin{gathered} endbeforeStart\left(task_{b,u1}, task_{b,u2}\right), \qquad \forall\, b \in Batches, \\ \forall\, u1 \in Units_{s1}, \forall\, u2 \in Units_{s2}, \forall\, s_1, s_2 \in Stages, s_1 + 1 = s_2 \end{gathered} \tag{3}$$

A efectos de restringir a uno la cantidad de *batches* que pueden ser procesados simultáneamente en un determinado equipo se plantean las Expresiones 4 y 5. En la primera de ellas se define la función acumulativa "*unitUsage*". Siendo cada unidad de procesamiento un recurso discreto unario, su capacidad máxima se fijó en uno y esa capacidad se considera "consumida" cada vez que una tarea es realizada en el equipo y liberada cuando la tarea

concluye. Para modelar esta situación se utilizó el constructor *"pulse(X, a)",* el cual toma el valor "*a*" durante el período que va desde el inicio hasta la finalización de la variable de intervalo "*X*".

$$unitUsage_u \leq 1, \forall\, u \in Units \tag{4}$$

$$unitUsage_u = \sum_{b \in Batches} pulse\left(task_{b,u}, 1\right), \forall u \in Units \tag{5}$$

La Expresión 6 permite representar las restricciones topológicas de la planta y la Expresión 7 los tiempos de *changeover*.

$$\begin{gathered} presenceOf\left(task_{b,u1}\right) \Rightarrow presenceOf\left(task_{b,u2}\right) = 0 \\ \forall\, b \in Batches, \forall u1 \in Units, \forall u2 \in F_{u1} \end{gathered} \tag{6}$$

$$\begin{gathered} presenceOf\left(task_{b1,u}\right) \cdot presenceOf\left(task_{b2,u}\right) \cdot \\ \min\left(endOf\left(task_{b1,u}\right) + co_{b1,b2} + su_u\,, endOf\left(task_{b2,u}\right) + co_{b2,b1} + su_u\,\right) \\ \leq \max\left(startOf\left(task_{b1,u}\right), startOf\left(task_{b2,u}\right)\right) \\ \forall\, b1, b2 \in Batches, \forall u \in Units \end{gathered} \tag{7}$$

Para poder modelar las secuencias de procesamiento prohibidas se recurrió a la definición de otra función acumulativa denominada *forbiddenChangeOver*, y al constructor *alwaysIn(cumulFunction, Y, a)* el cual obliga a la función *cumulFunction* a tomar el valor "*a*" durante el intervalo de tiempo que representa la variable de intervalo "*Y*". Dichas expresiones participan de las Restricciones 8 y 9, que son las que modelan las secuencias prohibidas.

$$alwaysIn\left(forbiddenChangeOver_{f.b2\,,u}, task_{f.b2,u}, 0\right), \qquad \forall\, f \in Fb, \forall u \in Units \tag{8}$$

Para definir la función acumulativa se utilizó el constructor *stepAtStart(X, a, b)* que trabaja de manera similar al constructor *pulse(X, a)* ya introducido; la diferencia aquí es que este constructor adopta un valor entre "*a*" y "*b*" cuando comienza el intervalo de la variable "*X*", pero no vuelve a cero cuando éste termina.

$$forbiddenChangeOver_{f,u} = stepAtStart\left(task_{f.b1,u}, 1{,}1\right) - \tag{9}$$

$$\sum_{b \notin f} stepAtStart\left(task_{b,u}, 0{,}1\right), \forall\, f \in Fb, b \in Batches, \forall u \in Units$$

La Expresión 10 permite contemplar las fechas de disponibilidad de equipos y *batches* y los *setup-times* de las unidades.

$$startOf\left(task_{b,u}\right) \geq \max(rd_u + su_u\ , rt_b)\,.\,presenceOf\left(task_{b,u}\right)$$
$$\forall\, b \in Batches, \forall\, u \in Units \qquad (10)$$

El modelo hasta aquí presentado permite abordar el problema de *scheduling* predictivo en una planta *batch* multiproducto multietapa, bajo diferentes políticas de almacenamiento intermedio/operación (UIS, NIS/ZW y NIS/UW). No obstante, el único recurso discreto que se contempla en el modelo es el de las unidades de producción. En la siguiente sub-sección se presentan extensiones que permiten contemplar otros tipos de recursos disponibles en cantidades limitadas.

3.3 Extensiones al modelo básico

Caso I. Esta extensión incorpora el modelado de otros recursos renovables diferentes de los equipos, tales como mano de obra, servicios, etc., que también son utilizados por las actividades de producción y cuya disponibilidad es acotada. Éstos se consideran recursos discretos (capacidad mayor o igual a uno) y para representar su empleo se definió la función acumulativa *resourceUsage* que participa en las Restricciones 11 y 12. En la definición de esta función acumulativa, y al igual que cuando se modeló la disponibilidad de equipos, se utilizó el constructor *"pulse(X, a)"*.

$$resourceUsage_r \leq avail_r, \quad \forall\, r \in Resources \qquad (11)$$

$$resourceUsage_r = \sum_{\forall\, b \in Batches} \sum_{\forall\, u\, Units_s} pulse\left(task_{b,u}, requir_{b,s,r}\right), \quad \forall\, r \in Resources \qquad (12)$$

Caso II. La segunda extensión contempla el modelado de la situación en la que la limpieza de un determinado equipo, entre dos *batches* de diferentes productos (*changeover*), se realiza

utilizando un recurso que posee una disponibilidad acotada. Para representar esta actividad de limpieza se recurrió a una nueva variable de intervalo denominada $cleanTask_{b1,b2,\ u}$. La duración de este tipo de tarea se asocia mediante la Expresión 13 a la duración del *changeover*, que fuera representado en la Expresión 7.

$$sizeOf\left(cleanTask_{b1,b2,u}\right) = co_{b1,b2} \cdot presenceOf\left(cleanTask_{b1,b2,u}\right) \quad \forall\, b1, b2 \in Batches, b1 \neq b2, \forall\, u \in Units \tag{13}$$

Debido a que una actividad de limpieza correspondiente a un cierto *changeover* entre dos tareas se ejecuta sólo si las mismas se llevan a cabo consecutivamente, en la Expresión 14 se recurre a la utilización de la variable de secuencia *taskSequence* y el constructor *typeOfNext* para su modelado.

La variable *taskSequence* establece una secuencia con las variables de intervalo que representan tareas ejecutadas en la unidad *u*, y el constructor *typeOfNext* establece qué variables no pueden estar una a continuación de la otra en dicha secuencia. Finalmente, para asegurar que el orden de las variables de intervalo en la secuencia se corresponda con el orden de ejecución de las tareas se utiliza el constructor *noOverlap*, tal como se indica en la Expresión 15.

$$presenceOf\left(cleanTask_{b1,b2,u}\right) = 0 \Rightarrow typeOfNext\left(taskSequence_u, task_{b,u}\right) \neq b2 \quad \forall\, b, b1, b2 \in Batches, \forall\, u \in Units \tag{14}$$

$$noOverlap(taskSequence_u, co), \qquad \forall\, u \in Units \tag{15}$$

La presencia de tareas de limpieza obliga a su sincronización con las actividades de procesamiento de los *batches*. Las Expresiones 16 y 17 son las que modelan la articulación entre estos dos distintos tipos de tareas. Así, si se realiza un *changeover* en un dado equipo, entre las actividades correspondientes al procesamiento de dos *batches* distintos, la variable de intervalo *cleanTask* deberá ubicarse temporalmente luego que haya finalizado la primera tarea y antes que se inicie la segunda.

$$endOf\left(task_{b1,u}\right) \cdot presenceOf\left(cleanTask_{b1,b2,u}\right) \leq startOf\left(cleanTask_{b1,b2,u}\right), \forall\, b1, b2 \in Batches, b1 \neq b2, \forall\, u \in Units \tag{16}$$

$$startOf(task_{b2,u}) \cdot presenceOf(task_{b1,u}) \geq endOf(cleanTask_{b1,b2,u}), \forall\, b1, b2 \in Batches, b1 \neq b2, \forall\, u \in Units \quad (17)$$

En virtud que las actividades de limpieza se llevan a cabo por medio de un número limitado de unidades destinadas a este fin, se definió en la Expresión 18 la función acumulativa *cleaningResourceUsage*, que modela el empleo de este recurso escaso por parte de las tareas de limpieza. En su definición se utilizó nuevamente el constructor *"pulse(X, a)"*. La Expresión 19 permite acotar el empleo simultáneo de los equipos de limpieza.

$$cleaningResourceUsage = \sum_{u \in Units} \sum_{b1,b2} pulse(cleanTask_{b1,b2,u}, 1) \quad (18)$$

$$cleaningResourceUsage \leq Qcr, \forall\, u \in Units \quad (19)$$

Finalmente, deben definirse las restricciones que corresponden a la función objetivo seleccionada. En el caso del *Makespan*, la Expresión 20 es la que la define.

$$endOf(Task_{b,u}) \leq Mk, \;\; \forall\, b \in Batches, \; \forall\, u \in Units \quad (20)$$

Con la consideración de las nuevas expresiones presentadas en la sección 3.3, el modelo puede contemplar la presencia de otros recursos discretos, disponibles en cantidades limitadas, que sean diferentes de los equipos. Inclusive, se pueden modelar los recursos de limpieza utilizados durante los *changeovers*. La incorporación de estas nuevas capacidades no requiere modificar la primera parte del modelo. Esta situación evidencia la flexibilidad y expresividad del modelo, y del lenguaje, para incorporar nuevos aspectos.

4 RESULTADOS

Se seleccionaron casos de estudio abordados en los trabajos de Jain y Grossmann (2001), Marchetti y Cerdá (2009), Zeballos, Novas y Henning (2011) para comprobar el desempeño de la propuesta descripta en la sección previa; éstos presentan variantes en cuanto al conjunto de aspectos contemplados y la función objetivo seleccionada.

Caso de estudio 1: Corresponde a un conjunto de problemas resueltos en el trabajo de (ZEBALLOS; NOVAS; HENNING, 2011). Se trata de una planta *batch* que posee 5 etapas de procesamiento y 25 unidades productivas. En cada etapa hay equipos no idénticos

operando en paralelo, y existen asignaciones producto-unidad prohibidas. Se consideran restricciones de distinta índole, entre las que se destacan: las topológicas (unidades de etapas sucesivas no conectadas entre sí), las vinculadas a secuencias de procesamiento prohibidas, de existencia de tiempos de limpieza entre *batches* (*changeover-times*) que son dependientes de la secuencia, tiempos de alistamiento o *setup* de los equipos, así como *ready-times* y *release-times* de unidades y *batches*. Además, se consideran tres escenarios diferentes, variando la política de almacenamiento intermedio/operación entre etapas: UIS, NIS/ZW y NIS/UW. Cada uno de los escenarios mencionados, a su vez, fue validado con diferentes números de *batches* (5, 12 y 22), generando así los 6 casos de estudio que se presentan en la Tabla 1. Los mismos se referencian de acuerdo al número de *batches* y de política adoptada; por ejemplo, el caso de 5 *batches* y política UIS se referencia como 5-UIS. El objetivo perseguido al generar las diferentes agendas de trabajo siempre se refiere a la minimización del *Makespan*.

Caso de estudio 2: Éste corresponde al ejemplo 4 del trabajo de Marchetti y Cerdá (2009). En la Tabla 1, este caso se identifica como P2. Se trata del problema de *scheduling* de un ambiente productivo conformado por 5 etapas y 12 equipos. Algunas de estas unidades operan en paralelo y no son idénticas entre sí. Asimismo, entre algunos equipos hay restricciones topológicas y también existen asignaciones *batch*-unidad prohibidas. Se consideran también tiempos de *setup* y *changeover*. Se contemplan tres tipos diferentes de recursos discretos (electricidad, vapor y recursos humanos) disponibles en cantidades limitadas, los cuales son demandados por ciertos *batches*, sólo en algunas etapas, y con diferentes requerimientos en cada caso. Por ejemplo, el *batch* 5 requiere, para su procesamiento, 8 unidades de vapor en la etapa 3 y 6 unidades en la etapa 5; en tanto el *batch* 7 necesita 7 y 6 unidades de vapor en estas mismas etapas y, además, demanda 7 unidades de electricidad en la etapa 3. El objetivo perseguido es la minimización de la tardanza total para una agenda de producción de 12 órdenes. Para abordar este caso de estudio se incorporó a la propuesta presentada en la Sección 3, el modelado de la tardanza de cada *batch* y una función objetivo que permite representar la tardanza total, como la sumatoria de las tardanzas individuales. La tardanza de cada *batch* es cero en caso que éste finalice a tiempo, o antes de su *due-date*, o, por el contrario, es igual a la diferencia entre su tiempo de terminación y su *due-date*.

Caso de estudio 3: Corresponde a los problemas descriptos en las tablas 16 y 17 del trabajo de Jain y Grossmann (2001); en la Tabla 1 estos dos casos se referencian como P3.a y P3.b,

respectivamente. Se trata de dos conjuntos de datos distintos para el mismo escenario: una planta que posee 3 etapas de producción y 8 equipos. Nuevamente, en cada etapa hay equipos no idénticos operando en paralelo, entre los que existen algunas restricciones topológicas. Asimismo, se presentan asignaciones *batch*-unidad prohibidas. Se consideran también tiempos de *setup* y costos. Éstos involucran tanto los de operación de equipos, independientemente del *batch* que éstos procesen, como los de manufactura de los *batches*, que sí dependen del equipo asignado. El objetivo es minimizar los costos totales. El modelado de esta función objetivo no se incluye por razones de espacio.

Los resultados de los tres casos de estudio se presentan en la Tabla 1. Los mismos corresponden a la solución de los diferentes modelos en una PC con 4GB de RAM, procesador *Intel*(R) *Core(TM) i5-2450M*, utilizando la estrategia de búsqueda "*reset*" que provee el *solver* de IBM-ILOG.

Tabla 1 – Resultados computacionales del modelo

Problema	**Resultados de otros autores**				**Resultados del presente trabajo**			
	Primera solución		Mejor solución		Primera solución		Mejor solución[a]	
	F.O.	CPU	F.O.	CPU	F.O.	CPU	F.O.	CPU
5-UIS	228.00	0.03	199.00	8.70	199.00	0.34	199.00	0.51
5-NIS/ZW	228.00	0.03	199.00	9.31	200.00	0.43	199.00	0.59
5-NIS/UW	228.00	0.03	199.00	10.31	206.00	0.35	199.00	0.54
12-UIS	354.00	0.11	293.10[b]	42.88	264.00	3.40	200.00[b]	309.78
12-NIS/ZW	381.00	0.13	311.20[b]	168.06	245.00	1.80	199.00[b]	132.57
12-NIS/UW	370.10	0.13	301.20[b]	378.00	245.00	1.54	199.00[b]	127.42
22-UIS	534.40	0.33	509.40[b]	4.47	273.00	3.38	291.00[b]	833.63
22-NIS/ZW	-	-	-	-	377.00	24.72	311.30[b]	578.43
22-NIS/UW	592.40	1.84	550.40[b]	26.63	352.90	45.36	352.90[b]	45.36
P2	-	-	31.6	114.05	93.40	2.24	31.60[b]	47.00
P3.a	-	-	111[b]	c	134	1.06	111[b]	19.81
P3.b	-	-	704[b]	c	766	0.17	704[b]	41.54

a. Solución obtenida en un tiempo máximo de 900 segundos de CPU.
b. Solución sub-óptima /solución óptima no comprobada.
c. Tiempos no reportados por diferencias tecnológicas en los equipos de cómputo.

Un análisis de los resultados obtenidos permite concluir que se obtuvieron las mismas soluciones reportadas por otros autores, o soluciones de mejor calidad, en tiempos de CPU reducidos. Sin embargo, dentro de los 900 s de CPU establecidos como límite, sólo fue posible garantizar que las soluciones halladas son las óptimas para los problemas de menor dimensión. También es posible apreciar que se obtuvieron soluciones iniciales de excelente calidad en muy bajos tiempos de CPU.

En la Tabla 2 se muestran los resultados obtenidos para diferentes variantes del problema 5-UIS introducido anteriormente en el Caso de estudio 1, con política de almacenamiento intermedio/operación UIS y 5 *batches* de diferentes productos a ser procesados. Las variantes permiten apreciar el impacto de un incremento de la dimensionalidad del problema al aumentar el número de *batches* de un mismo producto que son demandados; por ejemplo las variantes V3-V5 requieren 4 *batches* del producto 1 y 6 del producto 4. También posibilitan considerar escenarios de distinta complejidad al contemplar de forma explícita las actividades de limpieza y la existencia de un número creciente de equipos destinados a este fin. Los resultados obtenidos muestran el mayor esfuerzo computacional que es demandado al tener que resolver problemas de mayor tamaño.

Tabla 2 – Resultados computacionales de distintas variantes del problema 5-UIS

Variantes de 5-UIS	**Número de *batches* por producto**	**Cantidad de equipos de limpieza**	**Primera solución**		**Mejor solución[a]**	
			F.O.	CPU	F.O.	CPU
V1	[1,1,1,1,1]	1	344.30	19.67	199.00	2.12
V2	[2,1,1,1,1]	1	200.00	2.90	199.00	5.88
V3	[4,1,1,6,1]	No se contempla	297.50	0.56	212.60[b]	47.31
V4	[4,1,1,6,1]	1	212.60	138.31	212.60[b]	138.31
V5	[4,1,1,6,1]	2	243.60	38.3	212.60[b]	364.51

a. Solución obtenida en un tiempo máximo de 900 segundos de CPU.
b. Solución sub-óptima /solución óptima no comprobada.

A efectos de mejorar el desempeño computacional del modelo, en el problema P2 se incluyó una restricción adicional que, en cada equipo, ordena los *batches* asignados al mismo por fecha de entrega creciente. Este tipo de restricción también había sido agregada por (MARCHETTI; CERDÁ, 2009). Asimismo, al resolver los ejemplos que corresponden a la segunda extensión del modelo se seleccionó una estrategia de instanciación de variables que emplea conocimiento del dominio de trabajo. Específicamente, se definió el siguiente orden de instanciación: $task_{b,u}$, $cleanTask_{b1,b2,u}$.

5 CONCLUSIONES

Se propuso un modelo CP novedoso para el problema de *scheduling* predictivo de una planta industrial *batch* multiproducto, multietapa, con tiempos de limpieza dependientes de la secuencia. Se contemplaron unidades disímiles operando en paralelo en cada etapa y limitaciones en la disponibilidad de diferentes recursos renovables. El modelo se evaluó mediante casos de estudio con distinto grado de complejidad.

La formulación desarrollada, que en el futuro se valorará de manera más exhaustiva, exhibió un comportamiento robusto y permitió la fácil incorporación de nuevas restricciones, por lo que se la considera altamente extensible y flexible. Asimismo, el modelo fue capaz de mantener un muy nivel de desempeño al trabajar con diferente número de *batches* y al incorporar nuevas restricciones y variables, necesarias para contemplar recursos renovables distintos a los equipos, por lo que se lo considera un enfoque robusto y escalable. En todos los ejemplos resueltos los resultados mostraron que es posible obtener soluciones óptimas/subóptimas de muy buena calidad en bajos tiempos de CPU. También pudo apreciarse que es posible adoptar distintas funciones objetivo sin perder eficiencia computacional. El último ejemplo, además, permitió comprobar otras de las ventajas del entorno de trabajo elegido ya que es posible establecer estrategias de inicialización de variables que favorezcan el desempeño computacional.

Como líneas de trabajo futuro se apunta a (i) ampliar el alcance del modelo para considerar recursos renovables discretos cuya disponibilidad varía en el tiempo, así como órdenes de producción que incluyen múltiples *batches* a ser producidos en modo "campaña", (ii) explorar estrategias de instanciación e inicialización de variables que mejoren la eficiencia computacional en ejemplos de mayor tamaño.

SHORT-TERM SSCHEDULING OF MULTIPRODUCT MULTISTAGE BATCH PLANTS: A NOVEL CP APPROACH

ABSTRACT: This contribution addresses the short-term scheduling problem of multiproduct multistage batch plants, taking into account the existence of dissimilar parallel equipment units at each stage. To this end, a novel Constraint Programming (CP) methodology is presented. It can easily handle different features as well as a variety of limiting resources found in industrial environments. The model was tested by means of several medium-size examples taken from literature. The obtained results are discussed; they demonstrate a very good computational performance since optimal/sub-optimal solutions were found in reduced CPU times.

Keywords: Predictive scheduling, Batch plants, Constraint Programming, Limiting resources.

REFERENCIAS

CASTRO, P.M.; GROSSMANN, I.E. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. **Industrial & Engineering Chemistry**, v. 44, n. 24, p. 9175-9190, 2005.

HARJUNKOSKI, I.; GROSSMANN, I.E. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. **Computers and Chemical Engineering**, v. 26, n. 11, p. 1533-1552, 2002.

IBM. **ILOG CPLEX Optimization Studio.** Disponible en: http://www-01.ibm.com/software/integration/ optimization/cplex-optimization-studio/.

JAIN, V.; GROSSMANN, I.E. Algorithms for hybrid MILP/CP models for a class of optimization problems. **INFORMS Journal of Computing**, v. 13, n. 4, p. 258-276, 2001.

MARAVELIAS, C.T. General framework and modeling approach classification for chemical production scheduling. **AIChE Journal**, v. 58, n. 6, p. 1812-1828, 2012

MARCHETTI, P.A.; CERDÁ, J. A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers. **Computers and Chemical Engineering,** v. 33, n. 4, p. 871-886, 2009.

MÉNDEZ, C.A.; CERDÁ, J.; GROSSMAN, I.E.; HARJUNKOSKI, I.; FAHL, M. State-of-the-art review of optimization methods for short-term scheduling of batch processes. **Computers and Chemical Engineering** v. 30, p. 913-946, 2006.

ZEBALLOS, L.J.; NOVAS, J.M.; HENNING, G.P. A CP formulation for scheduling multiproduct multistage batch plants. **Computers and Chemical Engineering**, v. 35, p. 2973-2989, 2011.

Originais recebidos em: 17/09/2012

Aceito para publicação em: 02/05/2014