Enfoque estocástico novedoso: *scheduling* de plantas *batch* multiproducto, multietapa con incertidumbre en los tiempos de procesamiento

Franco M. Novara#*, Gabriela P. Henning#*

#Facultad de Ingeniería Química, UNL, Santiago del Estero 2829, 3000 Santa Fe, Argentina
*INTEC (UNL, CONICET), Güemes 3450, 3000 Santa Fe, Argentina

fnovara@fiq.unl.edu.ar,ghenning@intec.unl.edu.ar

Abstract. Existen numerosas metodologías de optimización determinísticas aplicadas a problemas de *planning* y *scheduling* de plantas industriales; pero son muchos menos los aportes que consideran la naturaleza estocástica de los datos empleados. Ante pequeñas variaciones en los mismos, las soluciones obtenidas pueden deteriorarse rápidamente, llevando a resultados de muy baja *performance*. En este trabajo se aborda el problema de *scheduling* de plantas *batch*, multiproducto, multietapa, en presencia de incertidumbre en los tiempos de procesamiento. Se desarrolla un modelo de programación con restricciones (CP) obteniéndose soluciones de mayor calidad respecto de aquéllas resultantes de aplicar un enfoque CP determinístico.

Keywords: *Scheduling* estocástico - Programación con restricciones - Plantas *batch* multiproducto, multietapa – *End time subject to deviation*

1 Introducción

En ambientes productivos cada vez más complejos, con empresas participando en mercados altamente globalizados y competitivos, y clientes cada vez más exigentes, lograr una alta eficiencia de operación es de vital importancia. Esta eficiencia es la que permite, por un lado, reducir costes y, por otro, alcanzar los altos estándares de nivel de servicio que pretenden los clientes. En este contexto, resulta crucial disponer de metodologías de planificación de la producción a corto plazo apropiadas.

La comunidad académica ha trabajado durante décadas en el desarrollo de herramientas que permitan asistir en la generación estas agendas (*schedules*) [1]. Se han realizado variadas propuestas para diferentes tipos de ambientes industriales, basadas en modelos de programación lineal mixta entera, programación con restricciones, algoritmos genéticos, heurísticas, etc. No obstante, la aplicación práctica de gran parte de estas herramientas es limitada por diversos motivos: (i) no logran capturar de manera eficiente los múltiples aspectos operacionales presentes en los ambientes reales; (ii) el número de tareas implicadas en un plan de producción para un caso real resulta

prohibitivo dado el carácter combinatorio del problema de *scheduling*; y (iii) en general, se ignora el comportamiento estocástico de los parámetros de operación.

En este trabajo se aborda el problema de *scheduling* de corto plazo para un ambiente industrial de tipo *batch*, multiproducto, multietapa, en el cual los tiempos de procesamiento están sujetos a incertidumbre. Esta propuesta emplea aportes anteriores de tipo determinístico [2] en los cuales se hizo foco en la complejidad y dimensionalidad del problema. En la Sección 2 se describe el problema tratado, así como las suposiciones realizadas. En la siguiente sección, se presenta la metodología y el modelo CP generado. En la Sección 4 se discuten resultados computacionales. Finalmente, se presentan conclusiones y trabajos futuros en la Sección 5.

2 Descripción del problema

2.1 El problema de scheduling.

Dado un conjunto de *batches*, cada uno de los cuales corresponde a un cierto producto, el problema de *scheduling* en una planta multiproducto, multietapa, consiste en asignar cada *batch* a una unidad de producción en cada una de las etapas del proceso, estableciendo sus tiempos de inicio y fin de procesamiento. En este tipo de ambientes todos los *batches* deben atravesar el mismo conjunto de etapas, en las cuales existen equipos no idénticos operando en paralelo. Puede ocurrir, además, que ciertos productos no puedan ser procesados en determinadas unidades.

Respecto a las tareas realizadas en un mismo equipo, entre tareas consecutivas se debe realizar una limpieza y/o alistamiento (*changeover/set-up*) cuyo tiempo depende de la unidad y de la secuencia de productos. Debido a *changeovers* prolongados/costosos, podría ocurrir que ciertas secuencias de procesamiento estén prohibidas.

En cuanto a la concatenación de las tareas realizadas sobre un mismo *batch*, ésta depende tanto de la topología de la planta, como de la política de operación adoptada. En virtud que no necesariamente todos los equipos de una cierta etapa están conectados con todas las unidades de la siguiente, la ruta que seguirá el *batch* está restringida. Además, de acuerdo a los requerimientos de proceso, las tareas podrían tener que ejecutarse una a continuación de la otra, sin demoras ni esperas, o con esperas limitadas o ilimitadas. En el caso que sea posible una espera entre etapas, la misma está condicionada por la presencia de tanques de almacenamiento. En este trabajo se supone que la espera es posible, pero que no existen tanques de almacenamiento; por lo tanto, de producirse la misma, el *batch* debe esperar dentro de la unidad que lo procesó (política NIS/UW, *non-intermediate storage - unlimited wait*).

El objetivo es confeccionar una agenda de trabajo que minimice la tardanza en la terminación de los *batches* respecto de su fecha de entrega (*due-date*).

2.2 Comportamiento estocástico de los tiempos de producción.

Los tiempos de procesamiento constituyen una de las principales fuentes de incertidumbre en el problema de *scheduling*. En general, considerar que la duración de una tarea de procesamiento no presenta variabilidad (es un parámetro constante) puede

conducir a agendas que exhiban una baja *performance* (un bajo desempeño en la práctica) o se tornen infactibles ante pequeños cambios en dichos tiempos.

Entre los principales enfoques estocásticos empleados en problemas de *planning* y *scheduling* se encuentran *stochastic programming*, *robust optimization* y *robust counterpart optimization* [3]. Otros autores han empleado enfoques alternativos basados en *fuzzy programming* [4] y en algoritmos genéticos [5]. En [6] puede encontrarse una revisión de los principales aportes en el área.

En este trabajo se utiliza un novedoso enfoque basado en CP, que a diferencia de la mayoría de las restantes contribuciones (a excepción de *robust counterpart optimization*) no emplea programación basada en escenarios.

3 Enfoque estocástico CP propuesto

La metodología de *scheduling* estocástico propuesta emplea un modelo CP para generar la agenda de trabajo. En éste se emplean variables que capturan lo que se denominará tiempo de finalización sujeto a desvío *(end time subject to deviation, et-StD)*, que está vinculado a una probabilidad de ocurrencia P. Para cada *batch b* procesado en la unidad *u, et-StD* en dicha unidad representa un tiempo que en el P% de los casos, al ejecutar la agenda, se espera sea mayor o igual al tiempo de finalización realmente acontecido para dicho *batch* en el equipo en cuestión.

Cada tarea de procesamiento, llevada a cabo en un cierto equipo sobre un determinado *batch*, posee un comportamiento estocástico y su duración está sujeta a variabilidad. Como consecuencia, el tiempo de finalización de una tarea está condicionado tanto por la variabilidad en la duración propia, como por la variabilidad en el tiempo de finalización de las tareas que se ejecutaron previamente y que podrían impactar en el tiempo de inicio del procesamiento. Como consecuencia, la variable que captura el tiempo de finalización de cada tarea es una variable aleatoria (VA). Si se conociera la distribución de probabilidad de cada una de estas VAs, podría establecerse el *et-StD* de cada *batch* en cada etapa, con una cierta probabilidad P.

Se supone que la duración de cada tarea se ajusta a una distribución normal, por lo que los tiempos de inicio y finalización de las tareas tendrán también un comportamiento similar (resultante de la suma de variables normales y tiempos de *changeover* y esperas que, en forma simplificada, se consideran sin variabilidad).

Para estimar el *et-StD* de un cierto *batch* en una determinada etapa, debe considerarse tanto el impacto de la variabilidad en las tareas antecesoras ejecutadas en la unidad a la que el *batch* fue asignado, como el impacto de las tareas predecesoras en las etapas previas, que pudieran condicionar el inicio en la etapa bajo consideración. Esto resulta muy costoso computacionalmente de capturar, por lo que se proponen dos alternativas simplificadas de estimación. A continuación éstas se presentan utilizando a modo de ejemplo el *schedule* de la Figura 1, que se supone conocido (ya generado).

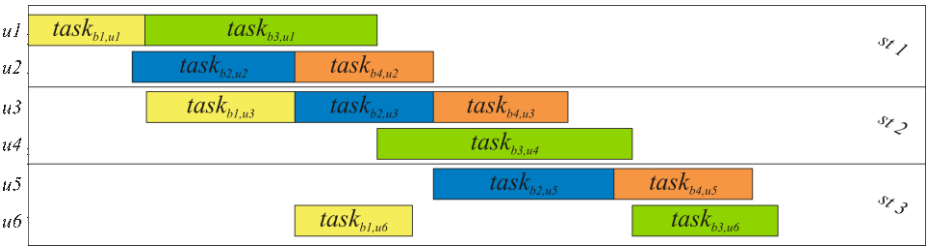

Fig. 1. *Schedule* sobre el que se ejemplifican las alternativas de estimación del *et-StD*.

3.1 Alternativa 1 para estimar el et-StD de una tarea.

Dado un *schedule*, se define $unEnd_{b,u}$, que estima el tiempo de finalización del *batch b* en la unidad *u*, sin considerar la posible influencia de las tareas realizadas en etapas previas sobre *b*. Se calcula a partir de los tiempos nominales o medios de las tareas predecesoras en la unidad *u* y el tiempo nominal del *batch b* en dicha unidad, más *n* veces el desvío estándar de esta VA, siendo *n* el número de desvíos estándar correspondientes a una probabilidad P. También puede calcularse $stEnd_{b,s}$ que estima el *et-StD* del *batch b* en la etapa *s*. En la etapa 1, para cada *batch b*, el valor de $stEnd_{b,s}$ es igual a $unEnd_{b,u}$, siendo *u* la unidad a la que fue asignado en dicha etapa. Para etapas posteriores a la primera (*s'*), $stEnd_{b,s'}$, toma el mayor valor entre $unEnd_{b,u}$ (*u* es la unidad a la cual se asignó el batch *b* en la etapa *s'*) y el de la estimación de *et-StD* en la etapa previa ($stEnd_{b,s}$, con $s+1=s'$), más el tiempo de procesamiento del *batch b* en la unidad *u,* más *n* veces el desvío estándar de dicho tiempo de ejecución.

En el cálculo de $unEnd_{b,u}$ se estima la varianza en el tiempo de finalización como la raíz cuadrada de las varianzas de los tiempos de procesamiento de los *batches* ejecutados previamente a *b* en la unidad *u,* incluyendo a *b*. Esta estimación es conservadora puesto que podrían existir *batches* entre los cuales haya una espera o tiempo muerto planificado que actúe como *buffer,* absorbiendo parte de esa variación. Además, debe notarse que este enfoque también es conservador al estimar los tiempos de finalización de un *batch* en etapas posteriores a la primera. Al tiempo de finalización en la etapa previa, se le suma la duración de la tarea en la etapa bajo análisis y *n* veces el desvío estándar de esa duración. Este desvío se combina linealmente con los desvíos de la etapa previa (no se realiza la combinación de varianzas), lo que sobreestima el tiempo de finalización.

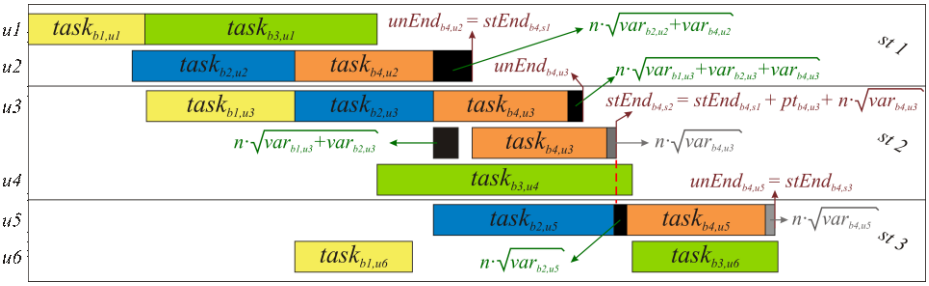

Fig. 2. Alternativa 1 para el modelado del tiempo de finalización sujeto a desvío de una tarea.

3.2 Alternativa 2 para estimar el et-StD de una tarea.

Dado un *schedule*, para cada *batch b,* asignado a una unidad *u*, puede calcularse $devUnitStart_{b,u}$, con el cual se estima el desvío estándar de la VA tiempo de inicio de dicho *batch* en esa unidad, sin considerar la posible influencia de las tareas realizadas en etapas previas sobre *b*. Su valor es igual a la raíz cuadrada de la suma de las varianzas de los tiempos de procesamiento de los *batches* ejecutados previamente a *b* en *u*. Luego, para cada *batch b*, se toma el mayor $devUnitStart_{b,u}$ denominado $devStart_b$. Además, dado un *schedule*, para cada *batch b,* puede determinarse $devBatch_b$, que corresponde a la raíz cuadrada de la suma de las varianzas de los tiempos de procesamiento de las tareas que requiere el *batch b*.

El *et-StD* para el *batch b*, en la última etapa *s'*, $stEnd_{b,s'}$, se estima como el tiempo esperado de finalización (calculado a partir de valores nominales), más *n* veces la cantidad $devStart_b$, más *n* veces la cantidad $devBatch_b$ (Figura 3).

Como se opera bajo una política NIS/UW, en el peor caso, la demora máxima $devStart_b$ identificada en la etapa *s* se trasmitirá a las etapas posteriores (*s+1*, *s+2*, etc.), a menos que exista una espera planificada que actúe como *buffer* de tiempo. Por el contrario, si aguas arriba existen demoras, al ser menores, serían "amortiguadas" por la demora proyectada $devStart_b$.

Con $devStart_b$ se busca desplazar todas las tareas del *batch b*, como un bloque, hacia adelante en el tiempo, producto de demoras en otros *batches*; mientras que $devBatch_b$ desplaza la finalización del procesamiento del *batch* como consecuencia de demoras en la propia ejecución.

Debe notarse que esta metodología es, además, conservadora porque suma linealmente $devStart_b$ y $devBatch_b$ para estimar el *et-StD*. También posee la misma característica conservativa que la alternativa anterior al calcular $unEnd_{b,u}$, en virtud que al determinar $devUnitStart_{b,u}$ se pasa por alto la existencia de tiempos muertos que pudiesen amortiguar/absorber las variaciones.

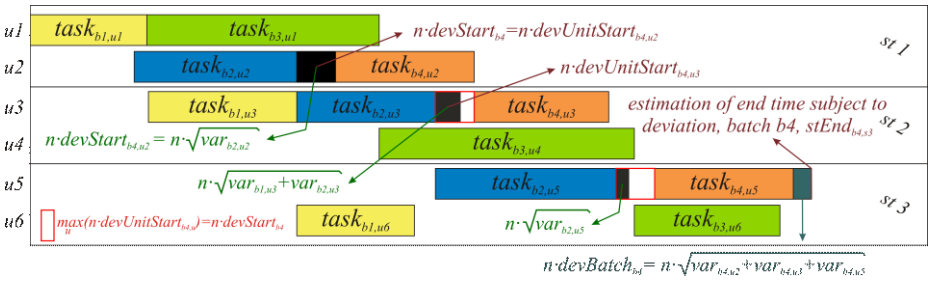

Fig. 3. Alternativa 2 para el modelado del tiempo de finalización sujeto a desvío de una tarea.

3.3 Modelo estocástico CP

El modelo presentado a continuación captura el problema de *scheduling* permitiendo generar la agenda de trabajo mediante ambas alternativas de estimación del *et-StD* propuestas. Sin embargo, anteriormente se supuso conocido el *schedule* motivo por el cual fue posible calcular el valor de los conceptos $stEnd_{b,s}$, $unEnd_{b,u}$, $devStart_b$ (deter-

minado a partir de máximo $devUnitStart_{b,u,}$ prescindiéndose de este último en adelante) y $devBatch_b$. En esta sección esos conceptos serán también variables del modelo.

Se realizó una implementación en el lenguaje OPL soportado por el paquete IBM ILOG CP *Optimizer* y el entorno de desarrollo IBM ILOG CPLEX *Optimization Studio* 12.5.1 [7]. Se ha optado por este ambiente por la eficiencia y *performance* de los constructores de alto nivel especialmente desarrollados para problemas de *scheduling*, que posibilitan manejar eficientemente las variables de intervalo, empleadas para representar cada tarea de procesamiento y la varianza de su duración, además de la elevada expresividad que puede lograrse con dichos constructores. Por otra parte, el lenguaje permite tratar los *changeovers* dependientes de la secuencia y del equipo sin incrementar el tamaño del modelo, ni los tiempos de resolución significativamente. Finalmente, brinda la posibilidad de emplear el denominado *warm start*, que permite suministrar una primera solución inicial reduciendo los tiempos de cómputo.

Nomenclatura.

Conjuntos/Índices.
B/b: *batches* requeridos en el horizonte de planeación.
C_u: unidades de la etapa posterior a la de la unidad *u* no conectadas con ésta última
F_p: productos que no es posible procesar inmediatamente después del producto *p*.
P/p: productos demandados en el horizonte de planeación.
S/s: etapas de procesamiento.
U/u: unidades de procesamiento.

Parámetros.
co: tripleta <*u,p,p'*> que contiene los tiempos de *changeover* entre un *batch* de producto *p* y otro de producto *p'* en la unidad *u*.
dd_b*:* fecha de entrega (*due-date)* del *batch b*.
n: número de desvíos estándar asociados a una probabilidad P fijada por el *scheduler*.
$pt_{p,u}$: tiempo de procesamiento de un *batch* de producto *p* en la unidad *u.*

Variables.
$devBatch_b$: variable flotante que representa el desvío estándar en el tiempo de procesamiento de cada una de las tareas que requiere el *batch b*.
devSeq: variable de secuencia definida para cada unidad *u*. Representa un ordenamiento de las variables de intervalo $var_{b,u}$ asociadas a la unidad *u*. Cada variable está caracterizada por un tipo que es igual al producto *p* al que corresponde.
$devStart_b$: var. flotante que representa el max. desvío estándar en el tiempo de proc. de las tareas que anteceden a la ejecución del *batch b* en las unidades asignadas.
$stEnd_{b,s}$*:* variable flotante, captura el *et-StD* de *batch b* en la etapa *s.*
$stTask_{b,s}$: variable de intervalo que representa la ejecución del *batch b* en la etapa *s.*
$task_{b,u}$: variable de intervalo que representa la ejecución del *batch b* en la unidad *u.*
$unEnd_{b,u}$*:* variable flotante, captura *et-StD* de *batch b* en la unidad *u.*
$unitBatchSeq_u$: variable de secuencia definida para cada unidad *u*. Representa un ordenamiento de las variables de intervalo $task_{b,u}$ asociadas a la unidad *u*. Cada tarea está caracterizada por un tipo que es igual al producto *p* al que corresponde.

$var_{b,u}$: variable de intervalo que representa la varianza del tiempo de procesamiento del *batch b* en la unidad *u.*

Restricciones.

El tiempo de inicio más temprano posible para el procesamiento de cada *batch* en cada unidad (mínimo entre el *ready-time* de la unidad y el *release-time* del *batch*), así como la duración de la tarea, pueden ser definidos al declarar la variable $task_{b,u.}$ Es posible establecer una duración mínima y máxima, así como un rango de tiempo en que la variable puede ubicarse, en caso de que esté presente en la solución. Esto constituye una de las ventajas del lenguaje OPL que permite mejorar la expresividad y reducir la cantidad de restricciones empleadas.

La restricción (1) establece que todo *batch* debe ser procesado en una y sólo una unidad por etapa, mientras que (2) establece la relación de precedencia entre las tareas realizadas sobre un mismo *batch* en diferentes etapas.

$$alternative\left(stTask_{b,s}, all(u \in U_s)\ task_{b,u}\right), \qquad \forall\, s \in S, \forall\, b \in B \tag{1}$$

$$\begin{gathered} endAtStart\left(stTask_{b,s}, stTask_{b,s'}\right) \\ \forall\, b \in B, \forall\, s, s' \in S, s \neq Card(S), s' = s + 1 \end{gathered} \tag{2}$$

Mediante (3) se evita que una misma tarea sea asignada a dos unidades de etapas sucesivas no conectadas entre sí, en cuyo caso al menos una de las dos variables que representan estas tareas tendrá duración nula (tiempo de terminación igual a cero).

$$\begin{gathered} minl\left(endOf\left(task_{b,\mathrm{u}}\right), endOf\left(task_{b,\mathrm{u}'}\right)\right) = 0 \\ \forall\, b \in B, \forall\, \mathrm{u}' \in \mathrm{C}_{\mathrm{u}}, \forall\, u, \mathrm{u}' \in U \end{gathered} \tag{3}$$

La restricción (4) permite insertar los tiempos de *changeover* entre tareas sucesivas, a la vez que impide que dos tareas realizadas en un mismo equipo se ejecuten con algún solapamiento. Por su parte, la expresión (5) evita la presencia de secuencias de procesamiento prohibidas. Ambas emplean variables de secuencia, cuyos detalles se presentaron oportunamente en [2] y se describen en [7].

$$noOverlap(unitBatchSeq_u, \mathrm{co}), \qquad \forall\, u \in U \tag{4}$$

$$typeOfNext\left(unitBatchSeq_u, task_{b,u}\right) \neq p', \forall (p, p')\ \in Fp, \forall\, u \in U, \forall\, b \in B_p \tag{5}$$

Mediante (6) se define $unEnd_{b,u}$; mediante (7) y (8), $stEnd_{b,s}$, para la etapa 1 y las etapas posteriores, respectivamente. Todas estas restricciones se incluyen cuando se opta por la alternativa 1 de estimación del *et-StD* de una tarea. Si se emplea la alternativa 2, (6) a (8) deben ser reemplazadas por (9) que captura $devStart_{b,u}$, (10) que captura $devBatch_b$ y (11) que establece el *et-StD* de cada *batch b* en la última etapa. Finalmente (12) introduce la función objetivo que busca minimizar la tardanza total a través de la minimización de la sumatoria de las tardanzas del *et-StD* respecto del *due-date* para cada *batch.*

$$unEnd_{b,u} = endOf\left(task_{b,u}\right) + n \cdot \sqrt{endOf\left(var_{b,u}\right)}, \forall b \in B, \forall\, u \in U \tag{6}$$

$$stEnd_{b,s} = \max_{\forall\, u \in U_s} \left(unEnd_{b,u}\right), \qquad \forall b \in B, \forall\, \mathrm{s} \in S, s = 1 \tag{7}$$

$$stEnd_{b,s'} = \max \left[\max_{\forall\, u \in U_{s'}} \left(unEnd_{b,u}\right), stEnd_{b,s} + sizeOf\left(stTask_{b,s'}\right) + n \cdot \sqrt{\max_{\forall\, u \in U_{s'}} \left(sizeOf\left(var_{b,u}\right)\right)} \right] \tag{8}$$

$$\forall\, b \in B, \forall\, \mathrm{s}, \mathrm{s}' \in S, s + 1 = s', s' > 1$$

$$devStart_b = \sqrt{\max_{\forall\, u \in U} \left(startOf\left(var_{b,u}\right)\right)}, \forall\, \mathrm{b} \in B, \forall\, \mathrm{u} \in U \tag{9}$$

$$devBatch_b = \sqrt{\sum_{\forall\, u \in U} sizeOf\left(var_{b,u}\right)}, \qquad \forall\, b \in B \tag{10}$$

$$stEnd_{b,s} = n \cdot devStart_b + endOf\left(stTask_{b,s}\right) + n \cdot devBatch_b \quad \forall\, b \in B, \forall\, \mathrm{s} \in S, s = card(s) \tag{11}$$

$$tardiness = \sum_{\forall\, \mathrm{b} \in B} maxl(0, stEnd_{b,s} - dd_b), \qquad \forall\, s \in, s = card(S) \tag{12}$$

Mediante (13) se establece, para cada tarea que forma parte de la solución, que la varianza asociada también pertenezca a la solución. La expresión (14) permite realizar el secuenciamiento de las variables de intervalo que representan la varianzas de los tiempos de procesamiento, para que sigan la misma secuencia que las tareas de procesamiento. Por su parte, (15) no permite solapamiento entre las variables de intervalo que representan varianzas en una misma unidad. De este modo, con los tiempos de finalización de las variables $var_{b,u}$ se puede conocer la estimación de la varianza de la VA asociada al tiempo de finalización del batch b en la unidad u.

$$presenceOf\left(task_{b,\mathrm{u}}\right) = presenceOf\left(var_{b,u}\right) \quad \forall\, b \in B, \forall\, u \in U \tag{13}$$

$$\min\left(endOf\left(task_{b',\mathrm{u}}\right) - endOf\left(task_{b,\mathrm{u}}\right), endOf\left(var_{b,u}\right) - endOf\left(var_{b',u}\right)\right) \leq 0, \qquad \forall\, b, b' \in B, \forall\, u \in U \tag{14}$$

$$noOverlap(devSeq_u), \qquad \forall\, u \in U \tag{15}$$

4 Resultados

La metodología propuesta se validó mediante la resolución de tres casos de estudio de *scheduling* determinístico presentes en la bibliografía, adaptados para incorporar incertidumbre en los tiempos de procesamiento. La política de operación considerada es de tipo NIS-ZW y el objetivo es reducir la tardanza total. Además, en todos los casos

se modeló la incertidumbre en los tiempos de procesamiento recurriendo a una distribución triangular. El tiempo de procesamiento determinístico original del ejemplo se tomó como la moda de la distribución. El valor mínimo se generó aleatoriamente a partir de dicha moda, restándole un porcentaje de hasta i*nf%* (es decir, se multiplicó el porcentaje i*nf%* por un número aleatorio diferente para cada tiempo, en el rango [0,1]). Similarmente, el valor máximo se generó adicionando a la moda un porcentaje aleatorio de hasta *sup%* dicho valor.

Si bien la metodología se construyó sobre la suposición de tiempos de procesamiento con distribución normal, por las ventajas que este tipo de distribución presenta al momento de combinar variables aleatorias, en la práctica son más frecuentes los atrasos que los adelantos en los procesamientos, por lo que las distribuciones son asimétricas. Por este motivo se decidió probar la metodología empleando estas distribuciones triangulares.

Caso 1. Corresponde al problema abordado por Marcheti y Cerdá en [8] (ejemplo 4). Se trata de un ambiente productivo conformado por 12 unidades de procesamiento, algunas de ellas operando en paralelo, pero no idénticas entre sí, distribuidas en 5 etapas. Se tienen en cuenta tiempos de *changeover* dependientes de la secuencia (se ignoran los tiempos de *set-up*). El objetivo es conformar la agenda de trabajo para 12 *batches* (12 productos diferentes) y minimizar la tardanza total de acuerdo a tiempos de entrega pre-establecidos. Respecto del caso de estudio original se realizaron las siguientes modificaciones: (i) se supone que el proceso opera bajo una política almacenamiento intermedio/espera de tipo NIS/ZW; (ii) se introducen restricciones topológicas; (ii) no se considera la presencia de recursos discretos de producción limitados (electricidad, vapor, etc.), (iv) se modifican las *due-dates* de las órdenes aleatoriamente para que exista al menos una solución determinística óptima (solución sin atraso).

Caso 2. Corresponde al abordado por [9]. Se cuenta con 25 unidades distribuidas en 5 etapas. Deben agendarse 22 *batches* de 22 productos diferentes. Se consideran *ready-times, release-times, set-up times, changeover times* dependientes de la secuencia, secuencias prohibidas y restricciones topológicas. Se emplean las *due-dates* correspondientes a "Set DD2", según referencia de autores.

Caso 3. Corresponde al ejemplo abordado en [10] (Problema 15). Deben agendarse 50 *batches,* de 50 productos diferentes, en un ambiente productivo que cuenta con 20 unidades distribuidas en 5 etapas. Sólo se consideran *changeover times* dependientes de la secuencia.

En la Tabla 1 se resumen los resultados obtenidos. Los problemas C1a a C1j corresponden a diferentes escenarios del primer caso de estudio, en los que se fue incrementando gradualmente la variabilidad en los tiempos de procesamiento. En todas las situaciones se empleó la alternativa 2 para estimar el *et-StD.* Con ella se obtuvieron mejores resultados que con la alternativa 1, los que por razones de espacio no se presentan. Los resultados correspondientes al segundo caso de estudio son C2a-C2b (alternativa 1) y C2c-C2d (alternativa 2). En los problemas C2b y C2d se recurrió al *warm start* utilizando como punto de arranque la solución determinística. Similarmente, los problemas C3a a C3d corresponden al tercer caso de estudio.

Tabla 1. Resultados de aplicar la metodología estocástica vs. la determinística a través de simulación.

Caso de estudio		inf	sup	Simulación solución estocástica**					Simulación solución determinística				
				Tardanza total	Nro. órdenes tardías	*Makespan*	Tiempo ocioso	Demora total tareas	Tardanza total	Nro. órdenes tardías	*Makespan*	Tiempo ocioso	Demora total tareas
C1a	A2	0,05	0,12	0,9	1	132,3	285,3	40,7	0,9	1	129,6	228,6	42,0
C1b	A2	0,075	0,18	1,4	1	128,6	270,2	62,0	1,4	1	130,6	233,7	66,1
C1c	A2	0,1	0,24	2,6	2	133,8	294,9	85,3	2,7	2	131,7	239,8	92,0
C1d	A2	0,125	0,3	4,0	2	133,0	288,2	109,7	4,2	2	132,9	247,1	122,0
C1e	A2	0,15	0,36	5,2	1	134,3	315,7	70,8	5,9	2	134,2	255,2	153,4
C1f	A2	0,175	0,42	5,7	1	139,7	265,9	117,6	7,5	2	135,5	262,9	185,1
C1g	A2	0,2	0,48	6,2	1	130,1	296,0	103,3	9,3	2	136,9	271,9	219,5
C1h	A2	0,225	0,54	6,6	1	138,5	310,3	129,7	10,9	2	138,3	279,9	250,1
C1i	A2	0,25	0,6	7,4	2	129,3	279,3	144,0	12,8	3	139,6	288,5	283,7
C1j	A2	0,275	0,66	8,8	3	140,1	359,1	157,3	16,5	5	141,0	297,1	317,8
C2a	A1	0,2	0,5	1,125	5	3,459	23,461	6,172	0,714	4	3,461	26,270	5,187
C2b*	A1	0,2	0,5	0,108	2	3,568	24,407	3,523					
C2c	A2	0,2	0,5	0,396	2	3,470	23,110	4,703					
C2d*	A2	0,2	0,5	0,060	2	3,440	23,767	3,122					
C3a	A1	0,07	0,2	75,2	5	616,9	2847,5	2679,7	73,4	4	619,3	3019,3	2350,9
C3b*	A1	0,07	0,2	58,9	4	612,2	2981,9	2295,1					
C3c	A2	0,07	0,2	101,0	5	621,6	2932,9	3066,2					
C3d*	A2	0,07	0,2	67,3	5	611,3	3055,8	2510,6					

*Se utiliza *warm start* - ** La solución determinística se obtiene mediante el modelo presentado en [2] – 4.500 segundos CPU – Equipo utilizado: Notebook Asus X555LAB, procesador Intel Core i7-5500U, Memoria Ram 8GB

En todas las situaciones, ya sea para el *schedule* obtenido por la metodología determinística o estocástica, se realizaron 50.000 simulaciones de ejecución, con las siguientes restricciones: (i) no adelantar ninguna tarea respecto de su tiempo de inicio planificado, en caso que la misma termine antes de lo previsto, y (ii) utilizar la estrategia de *rescheduling* "*right-shift*" en caso de demoras. Se reportan los valores promedios del total de simulaciones.

Puede concluirse que, en general, las soluciones del enfoque estocástico son mejores o comparables en relación a las del enfoque determinístico, **a pesar que para el problema 1 y 3 la solución determinística hallada fue óptima (tardanza cero).**

Mediante la alternativa 1 propuesta para el enfoque estocástico, considerando los resultados presentados en la Tabla 1 y otras pruebas realizadas, que aquí no se exponen por falta de espacio, no es posible obtener soluciones satisfactorias para casos de grandes dimensiones si no se recurre al *warm start* o a elevados tiempos de CPU. La alternativa 2 demostró ser más robusta y permite hallar soluciones en bajos tiempos de CPU, independientemente de que se adopte o no un *warm start*. No obstante, con cualquiera de las dos alternativas, con ambos tipos de inicialización, es posible hallar soluciones de buena calidad con bajo esfuerzo computacional.

Para el caso de estudio 1, la Figura 4 muestra la evolución de la función objetivo en relación con el aumento de la variabilidad en los tiempos de procesamiento. En todos los escenarios la tardanza fue menor o comparable para el enfoque estocástico. La Figura 5 muestra la evolución de la demora total en el inicio de las tareas programadas, siendo éste la suma de las demoras individuales del inicio de cada una con respecto a su inicio planificado. De la última figura se desprende que las soluciones encontradas mediante el enfoque estocástico son más estables y generarían menor "nerviosismo" en el momento de su ejecución en un ambiente industrial real.

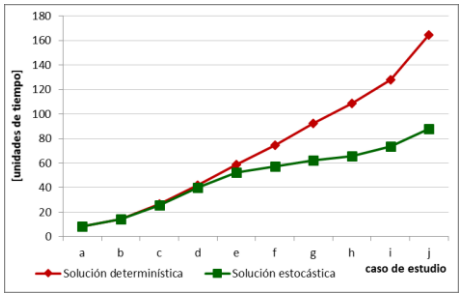

Fig. 4. Evolución de la tardanza total esperada con el aumento de la variabilidad en los tiempos de procesamiento (Caso 1, alternativa 2)

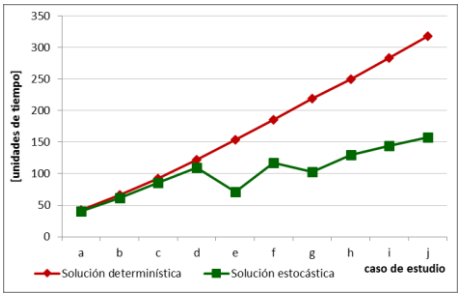

Fig. 5. Evolución del demora total en el inicio de las tareas con el aumento de la variabilidad en los tiempos de procesamiento (Caso 1, alternativa 2)

5 Conclusiones y comentarios finales

El enfoque propuesto constituye una alternativa a la utilización de escenarios y distribuciones discretas a las que suele recurrirse en otros aportes de modelado estocástico del problema de *scheduling*, lo cual incrementa significativamente la dimensiona-

lidad del modelo. En esta propuesta sólo se duplica el número de variables de intervalo que representan tareas, en contraposición a un enfoque de escenarios en el cual se multiplicarían por la cantidad de escenarios considerados y por la cantidad de combinaciones posibles entre las tareas. A conocimiento de los autores, éste sería el primer trabajo que emplea CP para abordar el problema estocástico de *scheduling*. Una limitación del mismo radica en la necesidad de aproximar cualquier distribución, que en la práctica siguiesen los tiempos de procesamiento, mediante distribuciones gausianas. No obstante, los ejemplos (en los cuales se adoptaron distribuciones asimétricas triangulares) muestran que el enfoque es lo suficientemente robusto para generar buenos resultados cuando los tiempos de procesamiento siguen otro tipo de distribución

Se plantea como trabajo futuro un análisis comparativo más sistemático de las dos alternativas estocásticas. Se tendrá en cuenta la incidencia del modelo en los tiempos ociosos que aparecen en el *schedule*. Asimismo, se extenderá el enfoque para contemplar políticas de operación/espera de tipo UIS (*unlimited intermediate storage*), NIS-ZW (*non-intermediate storage, zero wait*), NIS-FW (*non-intermediate storage, finite wait*), etc. Otra línea de trabajo estudiará el impacto de la calidad de la solución inicial utilizada en el *warm start*, en los tiempos de CPU y en la solución final.

6 Referencias

[1] Harjunkoski, I., Maravelias, C. T., Bongers, P., Castro, P. M., Engell, S., Grossmann, I. E., Hooker, J., Méndez, C., Sand, G., Wassick, J.: Scope for industrial applications of production scheduling models and solution methods *Comput. Chem. Eng.*, 62, pp. 161–193 (2014).

[2] Novara, F. M., Novas, M.J., Henning, G. P.: A novel constraint programming model for large-scale scheduling problems in multiproduct multistage batch plants: Limited resources and campaign-based operation," *Comput. Chem. Eng.,* 92, pp. 101-117 (2016).

[3] Li, Z, Ierapetritou, M. G.: Robust optimization for process scheduling under uncertainty, *Ind. Eng. Chem. Res.*, 47, 12, pp. 4148–4157(2008).

[4] Balasubramanian, J., Grossmann, I. E.: Scheduling optimization under uncertainty—an alternative approach, *Comput. Chem. Eng.*, 27, pp. 469–490 (2003).

[5] Bonfill, A., Espuña, A., Puigjaner, L.: Proactive approach to address the uncertainty in short-term scheduling, *Comput. Chem. Eng.*, 32, 8, pp. 1689–1706 (2008).

[6] Li, Z., Ierapetritou, M. G.: Process scheduling under uncertainty: Review and challenges, *Comput. Chem. Eng.*, 32, pp. 715–727 (2008).

[7] CPLEX - IMB ILOG - 12.5.1 http://www.ibm.com/support/knowledgecenter/SSSA5P_12.5.1/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html?lang=es.

[8] Marchetti, P. A., Cerdá, J.: A general resource-constrained scheduling framework for multistage batch facilities with sequence-dependent changeovers, *Comput. Chem. Eng.*, 33, 4, pp. 871–886 (2009).

[9] Zeballos, L. J., Novas, J. M., Henning, G. P. A CP formulation for scheduling multiproduct multistage batch plants, *Comput. Chem. Eng.*, vol. 35, no. 12, pp. 2973–2989 (2011).

[10] Castro, P. M., Harjunkoski, I., Grossmann, I. E.: Optimal short-term scheduling of large-scale multistage batch plants, *Ind. Eng. Chem. Res.*, 48, 24, pp. 11002–11016 (2009).