INGENIERÍA Y DESARROLLO

ARTÍCULO DE INVESTIGACIÓN / RESEARCH ARTICLE
http://dx.doi.org/10.14482/inde.34.1.7948

Algoritmo genético para solucionar el problema de dimensionamiento y programación de lotes con costos de alistamiento dependientes de la secuencia

Genetic algorithm for solving the lot-sizing and scheduling problem with sequence dependent setup costs

Iván Guillermo Peña-Arenas*
Corporación Universitaria Minuto de Dios, UNIMINUTO (Colombia)
Luis Francisco López-Castro**
Escuela Colombiana de Ingeniería Julio Garavito (Colombia)

* Magister en ingeniería industrial (Universidad de los Andes). Ingeniero industrial (Universidad Tecnológica de Pereira). Corporación Universitaria Minuto de Dios, Facultad de Ingeniería, Programa de Tecnología en Logística, Centro de Estudios Logísticos CELOG-MD. Profesor instructor. *ipena@uniminuto.edu*

** Magister en diseño y gestión de procesos (Universidad de la Sabana). Ingeniero industrial. Escuela Colombiana de Ingeniería Julio Garavito. Programa de Ingeniería Industrial. Profesor catedrático. *luis.lopezc@escuelaing.edu.co*

Correspondencia: Iván Guillermo Peña-Arenas. Corporación Universitaria Minuto de Dios (Sede Principal). Facultad de Ingeniería. Programa de Tecnología en Logística. Calle 81B No. 72B-70 Barrio Minuto de Dios, Bogotá D.C. (Colombia). Tel. (1)2916520 ext. 6894. *ipena@uniminuto.edu*

Volumen 34, n.º 1
Enero-junio, 2016
ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Fecha de recepción: 16 de agosto de 2014
Fecha de aceptación: 20 de noviembre de 2015

Resumen

El objetivo de este artículo es desarrollar un algoritmo genético el cual permita determinar los tamaños de lote de producción y su programación en un sistema de manufactura de una máquina para órdenes multiproducto, cuya función objetivo minimiza la suma de los costos de inventario por terminaciones tardías y de alistamiento. El problema contempla un conjunto de órdenes a ser procesadas con sus respectivas fechas de entrega. Cada orden debe ser entregada en su totalidad. Dentro de la programación de los trabajos se consideran tiempos de alistamiento dependientes de la secuencia. En la metaheurística implementada se utiliza de manera embebida un método heurístico para el cálculo de la función de adaptación. El método heurístico presentado es una variación del *Optimal Timming Algorithm* el cual involucra los tiempos de alistamiento dependientes de la secuencia. Se desarrolla un diseño de experimentos para probar el desempeño del algoritmo utilizando instancias generadas de forma aleatoria y comparando sus soluciones contra las encontradas por un método exacto. Los resultados muestran que el algoritmo logra un buen desempeño tanto en tiempo de ejecución como en calidad de la solución especialmente en instancias grandes.

Palabras clave: algoritmo genético híbrido, alistamiento dependiente de la secuencia, programación en una máquina, tamaño y programación de lotes.

Abstract

The main purpose of this paper is to develop a hybrid genetic algorithm in order to determine the lot sizes and their production scheduling in a single machine manufacturing system for multi-item orders, the objective function minimizes the sum of holding costs, tardy costs and setup costs. The problem considers a set of orders to be processed each one with its own due date. Each order must be delivered complete. In the scheduling are considered sequence dependent setup times. The proposed hybrid genetic algorithm has embedded a heuristic that is used to calculate its fitness function. The heuristic method presents a modification on the optimal timming algorithm in which are involved sequence dependent set up times. A design of experiments is developed in order to assess the algorithm performance, which is also tested using random-generated data and results are compared with those generated by an exact method. The results show that the algorithm achieves a good performance in both solution quality and time especially for large instances.

Keywords: Hybrid genetic algorithm, lot-sizing and scheduling, sequence dependent setup, single machine scheduling.

INTRODUCCIÓN

La programación (*scheduling*) es un proceso de toma de decisiones que busca asignar recursos escasos a tareas específicas en periodos de planeación determinados [1], dicha tarea es desafiante desde el punto de vista de las decisiones administrativas que conlleva debido a la complejidad de desarrollar algoritmos eficientes tanto en tiempo como en la calidad de la solución. La puesta en lotes (*batching* o *lot-sizing*) es el agrupamiento de ítems para transporte o procesamiento simultáneo, y que es a su vez un mecanismo que induce a una producción por fases que por lo regular no se sincroniza con la demanda o el consumo [2]. En industrias dedicadas a procesos que involucren lotes de producción, el tamaño de dichos lotes y el momento de procesarlos es una decisión importante en la que se debe considerar el cumplimiento de las órdenes, el costo de los inventarios, el tiempo y costo de los alistamientos, entre otros. Un problema básico de tamaño de lote es aquel en el que el tamaño de la orden de producción y su programación se determinan de tal forma que la demanda sea satisfecha a tiempo minimizando los costos totales de alistamiento y mantenimiento de inventarios [3]. Existen diversos estudios de revisión sobre tamaño y programación de lotes [2]-[8].

El problema de tamaño de lote ha sido modelado de dos formas diferentes: el de "ventana de tiempo grande" es el Problema de Programación y Tamaño de Lote Capacitado (*Capacited Lot-sizing and Scheduling Problem* o CLSP) que considera periodos largos de planeación en los cuales múltiples productos pueden ser manufacturados, el problema busca determinar cuánto producir en cada periodo de tal manera que se cumpla con las órdenes que tienen tiempos de entrega anteriores al final de dicho periodo, revisiones de este tipo de problemas se encuentran en [6] y [9]. El de "ventana de tiempo pequeña" es el Problema de Programación y Tamaño de Lote Discreto (*Discrete Lot-sizing and Scheduling Problem* o DLSP) que divide el horizonte de planeación en periodos más cortos de tal forma que solamente un producto pueda ser manufacturado en un único periodo [10]. Diferentes métodos de solución se han utilizado para la solución de este tipo de problemas. En [11] y [12] se proponen algoritmos híbridos en los cuales se utiliza una relajación langrangeana con el fin de descomponer el problema en sub-problemas, en el primer caso se resuelven utilizando el método *cross-entropy* y en el segundo un algoritmo de programación

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

dinámica. En [13] se propone un método exacto de solución, en donde en una etapa inicial se utiliza programación dinámica para obtener todas las secuencias eficientes para la programación de los trabajos y después se resuelve un problema de programación entera mixta el cual se encarga de la selección de la secuencia apropiada para un periodo determinado. Otros métodos heurísticos de solución se encuentran en [14] y [15].

El problema que integra las decisiones de determinar el tamaño de lotes de producción y de la programación de los mismos para varios productos en una máquina con capacidad finita es llamado Problema General de Programación y Tamaño de Lote (*General Lot-sizing and Scheduling Problem* o GLSP), este problema considera lotes continuos de producción para satisfacer demandas determinísticas dinámicas y tiene el objetivo de minimizar el costo de mantener inventarios y el costo de alistamientos dependientes de la secuencia [16]. Este tipo de alistamientos se desarrollan en [17]-[26]. Modificaciones a este tipo de formulación se han realizado al considerar faltantes en la función objetivo del problema y a su vez resolviendo de manera conjunta las decisiones sobre materia prima y tamaño del lote [27]. En [28] se cambia de manera radical la función objetivo pasando a una en la cual se maximizan las utilidades en lugar de minimizar los costos, además la restricción de demanda se relaja brindando la posibilidad de que sea satisfecha o no en cada periodo.

Dentro de los modelos de tamaño de lote se encuentran los llamados Problemas de Programación por Lotes (*Batching and Scheduling Problem* o BSP) [4] en los cuales se considera el comportamiento de la demanda como continuo. De esta forma cada demanda se encuentra caracterizada por el tipo de ítem y la cantidad de productos. Las demandas son entonces interpretadas como trabajos, en donde su tiempo de procesamiento es proporcional al tamaño (cantidad de productos) del trabajo.

De forma general se presentan diferentes consideraciones en el dimensionamiento y programación por lotes de producción. En algunos casos los lotes de producción ya se encuentran calculados y se resuelve la programación de los mismos sujetos a ventanas comunes de entrega asignables [29]. En [30] todos los trabajos tienen el mismo tiempo de producción, los tiempos de alistamiento se asumen idénticos para todos los lotes y su función objetivo es minimizar el máximo tiempo de terminación y el tiempo total de flujo.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Por el contrario, en [31] cada trabajo tiene su propio tiempo de producción, no se consideran tiempos de alistamiento y la función objetivo es la minimización del tiempo total de terminación. Otra consideración susceptible de ser realizada es el arribo dinámico de trabajos como se presenta en [32].

En [33] se resuelve el DLSP realizando una re-formulación del problema a partir del BSP, en ésta la demanda para un ítem la (interpretada como un trabajo) tiene tiempo de entrega y de procesamiento. Terminar los trabajos antes del tiempo de entrega se penaliza con los correspondientes costos de mantener en el inventario, no se permiten faltantes y se consideran costos y tiempos de alistamiento dependientes de la secuencia. De lo anterior se concluye que una solución satisfactoria para el DLSP pude ser hallada reformulando dicho problema como un BSP y que, además, este último bajo ciertas características podría ser entendido como un problema de secuenciación de trabajos en una máquina.

En el presente artículo se presenta un nuevo modelo el cual modifica el BSP clásico agregando penalizaciones por terminaciones tardías, al igual que el desarrollado en [3], pero adicionando la consideración de tiempos de alistamiento dependientes de la secuencia. A este se le ha denominado Problema de Secuenciación por Lotes de Producción con Penalizaciones por Terminaciones Tempranas, Tardías y de Alistamiento Dependientes de la Secuencia (*Batch Sequencing Problem with Earliness Tardiness* and *Setup Dependent Penalties* o BSPET).

Una alternativa para la solución de problemas complejos de optimización es combinar métodos de optimización tales como metaheurísticas, programación matemática, programación con restricciones y aprendizaje de máquina entre otros. La idea subyacente para utilizar la hibridación es la sinergia producida por explotar el carácter complementario de diferentes estrategias de optimización [34], una clasificación de metaheurísticas híbridas se muestra en [35]. Ejemplos de hibridación de métodos metaheurísticos para la solución de problemas de programación de operaciones puede verse en [36]-[38].

Para el caso de los Algoritmos Genéticos (*Genetic Algorithms* o GA), esta idea se plasma en los Algoritmos Genéticos Híbridos (*Hybrid Genetic Algorithm* o HGA) en los que se combinan algoritmos o técnicas de mejora con alguno de los operadores del GA con el fin de mejorar los individuos o facilitar

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

la velocidad de convergencia. Los HGA son esquemas de optimización cooperativa en el que los algoritmos incorporados trabajan en conjunto con los operadores genéticos [39]. Tanto los GA como los HGA han sido ampliamente utilizados como método de solución para problemas de tamaño de lote y de programación en una máquina y pueden verse [40]-[48].

El presente artículo tiene como objetivo desarrollar un HGA como método de solución al Problema de Secuenciación por Lotes de producción con penalizaciones por terminaciones Tempranas, Tardías y Alistamiento Dependiente de la Secuencia, el algoritmo propuesto utiliza una modificación de la heurística *Optimal Timing Algorithm* [41] para calcular la función de adaptación de cada individuo. En este trabajo se pretende probar la efectividad del HGA propuesto para la solución de este tipo de problemas de secuenciación de lotes de producción. La prueba se realiza utilizando instancias generadas de forma aleatoria por medio de un diseño experimental Taguchi y comparando el resultado contra el obtenido por un algoritmo de Ramificación y Corte (*Branch and Cut* o B&C).

METODOLOGÍA

En orden a desarrollar y probar el funcionamiento del algoritmo genético hibrido que se propone en el presente artículo a continuación se desarrollan las siguientes subsecciones: caracterización y formulación del problema, descripción del algoritmo genético y experimentos computacionales. En este último se propone un diseño de experimentos del tipo Taguchi el cual, a partir de un tamaño de muestra menor (comparado con un diseño factorial), nos permita inferir la eficiencia del algoritmo propuesto.

Características y formulación del problema

El Problema de Secuenciación por Lotes de producción con penalizaciones por terminaciones Tempranas, Tardías y Alistamiento Dependiente de la Secuencia (*Batch Sequencing Problem with Earliness Tardiness* and *Setup Dependent Penalties* o BSPET) consiste en determinar los tamaños de los lotes de producción y su programación, de forma tal que la suma de los costos de alistamiento, los costos de mantener unidades en el inventario, y los costos o penalizaciones por entregas tardías se minimice, sujeto a cumplir con las órdenes de producción en firme y los pronósticos de la demanda.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

El problema es una variación al BSP del cual el DLSP es un caso particular [33], ya que el BSP es *NP-Hard* [4] el BSPET también lo es.

El BSPET se caracteriza por: Tener un conjunto de trabajos a ser procesados en una sola máquina al inicio del horizonte de planeación. La demanda de cada orden de producción es convertida en lotes de producción o trabajos, en donde, cada *lote* es el número de unidades de tiempo que tarda la máquina (a toda capacidad) en producir el número de unidades de cada orden de producción. Un *bloque* es el conjunto de trabajos programados consecutivamente los cuales no se encuentran separados por tiempos ociosos. Los bloques pueden estar formados por trabajos que pertenecen a diferentes tipos de productos. Cada trabajo será despachado al cliente solamente en su fecha de entrega. Es decir, no se permiten entregas antes de la fecha pactada, y en el caso que el trabajo se termine después de la fecha de entrega éste será entregado justo en el momento en el que se termine.

Cada producto tiene una cantidad de trabajos $n_{i,}$ y el total de trabajos que se tienen al inicio para ser programados es igual a J, es decir, la suma de los n trabajos para cada producto i. Se colocan en paréntesis los atributos del trabajo (i, j) para denotar el trabajo como una sola entidad, es decir, el *j-ésimo* trabajo del producto i. Cada trabajo tiene asociado un tiempo de procesamiento $P(i, j)$, tiempo de alistamiento $St(i, g)$ por pasar de producir un trabajo del tipo de producto i a un trabajo del tipo de producto g, una fecha de entrega $d(i, j)$ y unos costos correspondientes a guardar esa cantidad de unidades en el inventario $h(i, j)$ por unidad de tiempo que son penalización por la terminación temprana del trabajo, costos por terminación tardía del trabajo $TC(i, j)$, y costos de alistamiento $SC(i, g)$ por pasar de producir un trabajo del tipo de producto i a un trabajo del tipo de producto g. Las variables de decisión $C(i, j)$, $T(i, j)$ y $E(i, j)$, son respectivamente el tiempo de terminación, la tardanza y la terminación temprana del trabajo (i, j). Estas variables dependen de la secuencia de trabajos $\pi = (i_{[1]}, j_{[1]}), (i_{[2]}, j_{[2]}),\dots, (i_{[k]}, j_{[k]}),\dots, (i_{[J]}, j_{[J]})$ donde $(i_{[k]}, j_{[k]})$ es el trabajo en la posición k. La formulación del BSPET se muestra a continuación:

Función objetivo:

$$Min\ Z(\pi) = \sum_{i=1}^{J} [h(i_{[k]}, j_{[k]}) \times E(i_{[k]}, j_{[k]}) + TC(i_{[k]}, j_{[k]}) \times T(i_{[k]}, j_{[k]}) + SC(i_{[k]}, j_{[k]})] \quad (1)$$

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Sujeto a las siguientes restricciones:

$$C(i_{[k-1]}, j_{[k-1]}) + P(i_{[k]}, j_{[k]}) + St(i_{[k]}, j_{[k]}) \leq C(i_{[k]}, j_{[k]}) \quad k = 1,\ldots,J \quad (2)$$

$$E(i_{[k]}, j_{[k]}) = max(0, [d(i_{[k]}, j_{[k]}) - C(i_{[k]}, j_{[k]})]) \quad k = 1,\ldots,J \quad (3)$$

$$T(i_{[k]}, j_{[k]}) = max(0, [C(i_{[k]}, j_{[k]}) - d(i_{[k]}, j_{[k]})]) \quad k = 1,\ldots,J \quad (4)$$

La función objetivo presentada en la ecuación (1) es la sumatoria de los costos de mantener inventario, penalización por terminaciones tardías de los trabajos y alistamiento. La ecuación (2) modela la restricción en la cual se obliga a que cada trabajo respete la secuencia y los tiempos de alistamiento, mientras que en las ecuaciones (3) y (4) se realiza el cálculo de las terminaciones tempranas o tardías de los trabajaos respectivamente dada la secuencia de programación π. En la práctica, para involucrar los pronósticos de la demanda dentro de la programación de los trabajos, se crean órdenes de producción ficticias para cada tipo de producto *i*, las cuales correspondan a los pronósticos de las demandas, con fecha límite igual a la fecha final para la cual se ha hecho el pronóstico. El tamaño de la orden será igual a la diferencia entre la sumatoria de la cantidad de productos del ítem *i* que se encuentra en el total de órdenes de producción en firme y el pronóstico de la demanda para dicho ítem.

Algoritmo genético

Debido a su naturaleza compleja (*NP-Hard*), el BSPET es susceptible de ser resuelto utilizando un método basado en GA o HGA. Los criterios para el uso de los GA como método de solución son presentados en [49]. En este artículo se presenta la hibridación entre un algoritmo genético y un algoritmo de propósito específico como lo es el *Optimal Timing*, en donde de acuerdo con la estrategia de control implementada, el segundo es un subordinado del primero. De esta forma se establece un nivel de hibridación de bajo nivel (*low level hybridization*) dado que la metaheurística asigna una de sus funciones, en este caso el cálculo de la función objetivo a la heurística. Esto configura una relación de fuerte dependencia entre ambos algoritmos. Este tipo de hibridación puede ser encontrada dentro de las clasificaciones

realizadas sobre metaheurísticas híbridas en [34] y [50]. A continuación se explican los operadores del HGA diseñado para el BSPET.

- **Representación**

Cada solución factible llamada cromosoma se representa por la secuencia de producción de los trabajos. No se hace necesario presentar la información de los bloques de trabajos dado que serán determinados en la evaluación de la función objetivo. Cada posición del cromosoma es un *gen* y el valor que toma es un *alelo*. La figura 1 muestra un ejemplo de representación para *J=6*.

Figura 1. Cromosoma para seis trabajos

- **Inicialización**

La población inicial se genera de forma completamente aleatoria [41] con tamaño de 100 individuos, este número permanece constante a lo largo de todas las generaciones hasta que se cumpla el criterio de detención [3]. En este caso el criterio de detención es el que ocurra primero entre: funcionamiento del algoritmo por 200 generaciones, o diez generaciones continuas sin cambios en la función objetivo.

- **Cruce**

El operador de cruce seleccionado es el Uniform Order Based Crossover [41], el cual se explica a continuación:

Primero utilizando el método de la ruleta se seleccionan dos soluciones (cromosomas) de la población y se genera de forma aleatoria una plantilla de ceros y unos.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

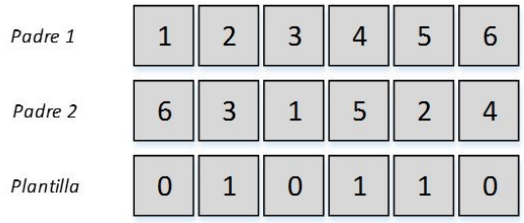

Figura 2. Padres y plantilla para cruce

Después se determinar los genes que pasan sin cambios a la descendencia. Para cada posición se revisa el valor de la plantilla, si el valor es cero (0), el gen del primer padre pasa directamente al primer hijo en la misma posición. Por el contrario, si el valor es uno (1), el gen del segundo padre pasa directamente al segundo hijo en la misma posición.

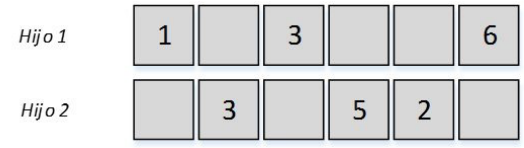

Figura 3. Descendencia con genes directos de los padres

Por último el conjunto de genes que requieren cambios del padre 1 pasan al hijo 1 llenando los genes vacíos en el orden en el que aparecen en el padre 2. De manera similar, el conjunto de genes que requieren cambios del padre 2 pasan al hijo 2 llenando los genes vacíos en el orden en el que aparecen en el padre 1.

Figura 4. Descendencia final

Este operador garantiza que todas las soluciones de la descendencia sean factibles, es decir, que no haya trabajos repetidos o sin programar.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

- **Mutación**

Se utiliza mutación por intercambio aleatorio en dos puntos, es decir, para un individuo se seleccionan aleatoriamente dos trabajos y se intercambia su posición. La probabilidad de mutación seleccionada P_c es de 0,6 [41].

- **Evaluación y selección**

Para el cálculo de la función de costos C_k de cada cromosoma k de la población se utiliza una modificación a la heurística *Optimal Timing Algorithm*. La cual, dada una secuencia de trabajos, localiza de forma óptima el tiempo de inicio cada trabajo j considerando su fecha de entrega [41]. Se toma la secuencia de trabajos y a partir de ella se determina el conjunto de bloques de trabajo $B_1, B_2, \ldots, B_n$ de la siguiente forma: el primer trabajo se programa de tal manera que se termine de fabricar en su fecha de entrega. El segundo trabajo y los trabajos subsiguientes se programan de la siguiente forma: Se programa el trabajo $k+1$, si $C(i_{[k]},j_{[k]}) + P(i_{[k+1]},j_{[k+1]}) + St(i_{[k]},j_{[k+1]}) \leq d(i_{[k+1]},j_{[k+1]})$, entonces el trabajo $k+1$ se terminaría de producir en su fecha de entrega y formaría un nuevo bloque, en caso contrario es programado para que empiece en $C(i_{[k]},j_{[k]})$ y forme parte del bloque al que pertenece el trabajo k. Después de determinar los *bloques* de trabajos, se debe calcular el tiempo de inicio de cada bloque. Los bloques de trabajos se empiezan a mover hacia la izquierda y en cada movimiento se calcula la función objetivo $G(x)$ del bloque, la cual es la suma de los costos en que se incurre debido al orden de la secuencia de trabajos. Los criterios de parada del movimiento son:

$$C(i_{[primero]},j_{[primero]}) - P(i_{[primero]},j_{[primero]}) = 0 \quad (5)$$

$$G(x\text{-}1) > G(x) \quad (6)$$

$$C(i_{[k]},j_{[k]}) + P(i_{[k]},j_{[k]}) + St(i_{[último]},j_{[k]}) = C(i_{[último]},j_{[último]}) \quad (7)$$

En el caso (5), el primer trabajo del bloque 1 (B_1) inicia en el momento cero del horizonte de planeación, en (6) el valor de la función objetivo del bloque se deteriora al ser movido una unidad de tiempo hacia la izquierda, lo cual significa que el mínimo valor de la función de costos de bloque ha

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

sido alcanzado, y en el caso (7) el bloque actual que se está moviendo hacia la izquierda se fusiona al bloque inmediatamente anterior.

El cálculo de la función de adaptación $fit(C_z)$ de cada individuo es muy importante en el algoritmo, dado que determina la calidad de las soluciones, buscando que los individuos con mejores funciones de costos sean los que tengan mayores probabilidades de ser seleccionados para ser cruzados y sean escogidos para ser parte de las generaciones posteriores. La función de aptitud para cada individuo de la población se calcula de la siguiente forma:

$$fit(C_z) = \frac{(C_{peor} + 1) - C_z}{\sum_{n=1}^{100} C_1} \quad (8)$$

En donde C_{peor} es la función de costos del peor individuo de la generación actual. En la selección de los individuos que pasarán de una generación a la siguiente, se utiliza una estrategia *elitista*, en donde pasa siempre el mejor individuo de la generación actual. El diagrama de flujo del algoritmo propuesto se muestra en la Figura 5.

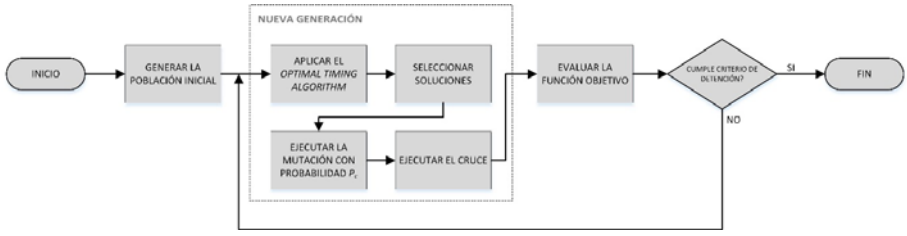

Figura 5. Diagrama de flujo del HGA propuesto

Experimentos computacionales

Se desarrolló el diseño de experimentos con el fin de evaluar el desempeño del HGA implementado siguiendo la metodología desarrollada en [3]. De esta forma, se determinó su funcionamiento bajo distintos tamaños de órdenes, diferentes costos por mantener unidades en el inventario, costos y tiempos de alistamiento y penalizaciones por terminaciones tardías de las órdenes de producción. El diseño de experimentos es del tipo Taguchi ortogonal L_{27} para tres factores y tres niveles generado por medio del software

Minitab 16, la cantidad de tratamientos y los respectivos niveles para cada factor son los que se muestran en la Tabla 2. La selección se sustenta en el hecho de que representa una considerable reducción de tiempo y costo de análisis al compararlo con un diseño factorial, debido a que se centra en el concepto de ruido y robustez en los procesos.

Para evitar las limitaciones del uso de un valor en específico se utiliza la proporción entre los diferentes costos [3]. Los factores a analizar son: **A** el tamaño del problema, es decir, la cantidad de órdenes de producción al inicio del periodo de planeación, **B** la proporción entre las penalizaciones por terminaciones tardías y el costo por mantener unidades en el inventario (TC_i/h_i), y **C** la proporción entre el costo de alistamiento y el costo por mantener unidades en el inventario (SC_{ij}/h_i). Los diferentes factores y sus niveles son mostrados en la Tabla 1.

Tabla 1. Factores y niveles del diseño de experimentos

Factor	Nivel		
	1	**2**	**3**
A	15	30	60
B	0,25	1	4
C	1	10	20

El factor A corresponde al número de trabajos a procesar, son de 15, 30 y 60 para los niveles 1, 2 y 3, respectivamente, la longitud o tiempo de proceso de cada orden de producción es generado utilizando una distribución uniforme discreta con parámetros 1 y 10. Tamaños de órdenes similares pueden encontrarse en instancias utilizadas en [48], [51]-[54].

De igual forma el tipo de producto es determinado mediante un número aleatorio generado de una distribución uniforme discreta entre 1 y 5, es decir, que hay cinco tipos de productos. Lo ajustado de la fecha de entrega para cada trabajo (i, j) es fijado con un factor de 1.2. La fecha de entrega para cada trabajo (i, j) es determinada utilizando la siguiente fórmula.

$$d(i, j) = U\left[P(i, j),\ 1.2 \times \sum_{k=1}^{J} P(i_{[k]}, j_{[k]})\right] \quad (9)$$

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

En donde la fecha de entrega del trabajo (i, j) es igual a un número aleatorio de una distribución uniforme discreta entre el tiempo de proceso $P(i, j)$ del trabajo y la multiplicación del factor de ajuste 1.2 por la sumatoria de todos los tiempos de proceso de los trabajos a ser programados al inicio del periodo de planeación.

La tasa del costo de inventario por unidad de tiempo del tipo de producto i es generada de una distribución uniforme continua con parámetros (0,10]. El valor de la tasa por penalización por terminaciones tardías del producto i es igual a $TC_i = B_l \times h_i$, donde B_l es el valor de factor B en el nivel l y C_l el valor de factor C en el nivel l, en donde l toma valores 1, 2 y 3. La tasa del costo por alistamiento al pasar de producir el producto i al producto g es igual a $Sc_{i,g} = C_l \times h_i$, la cual no depende de los tamaños de los lotes a producir. Para determinar los tiempos de alistamiento que se consideran dentro de la secuencia de programación de los trabajos, a cada factor C_l se le asigna una distribución uniforme discreta (entre 1 y 3, 2 y 4, 3 y 5 para C_1, C_2 y C_3, respectivamente), las cuales incrementan los tiempos de alistamiento conforme aumentan los costos por alistamiento con respecto a los costos por tener unidades en el inventario.

RESULTADOS

El HGA se desarrolló en lenguaje Java, las instancias se corrieron en un computador personal con procesador *AMD® Athlon® II X2 250 con 4 GB de memoria RAM*. Con el fin de probar el desempeño del método de solución propuesto se comparó con la solución alcanzada por el método B&C implementado en *CPLEX v.12.3.0* en el mismo equipo. Ya que la formulación presentada es no lineal, es necesario formular el BSPET como modelo de Programación Entera Mixta, dicha formulación se presenta en el anexo final. Para todas las instancias el tiempo de ejecución fue limitado a 7.200 segundos. Como se puede ver en la Figura 6 se ejecutó el B&C para instancias de prueba de 15, 30 y 60 trabajos, durante 7.2, 4.4 y 8.8 segundos encontrándose que no hay mejora significativa después de los 7200 segundos. El mismo tiempo límite de ejecución para comparación es utilizado en [41] y [23].

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

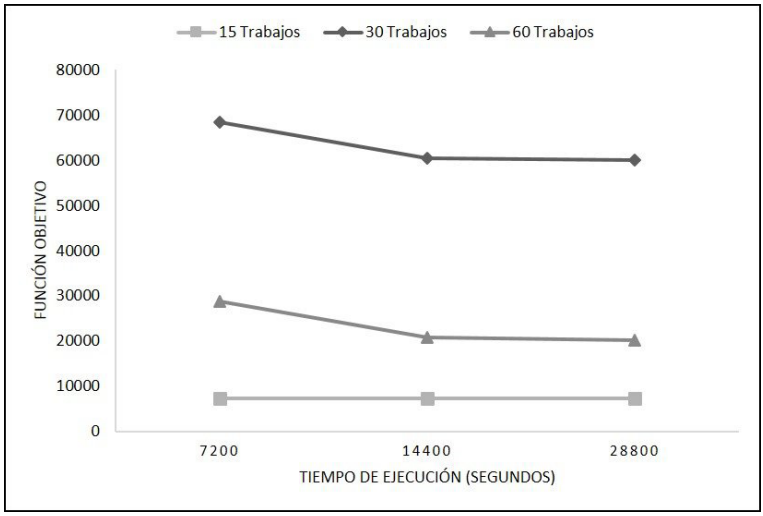

Figura 6. Desempeño del B&C para instancias de prueba

La Tabla 2 muestra los tratamientos que componen el experimento Taguchi, en este caso un tamaño de muestra correspondiente a 27 combinaciones. De igual forma, se presenta la diferencia porcentual (*%GAP*) que se calcula como:

$$\%GAP = \frac{FO_{HGA} - FO_{B\&C}}{FO_{B\&C}} \times 100 \qquad (10)$$

En donde FO_{HGA} y $FO_{B\&C}$ son los resultados de la función objetivo obtenidas con el algoritmo propuesto y el método exacto respectivamente.

Tabla 2. Tratamientos y %GAP

Tratamiento	Factor			%GAP
	A	B	C	
1	1	1	1	0,01%
2	1	1	1	0,00%
3	1	1	1	0,00%
4	1	2	2	0,00%
5	1	2	2	0,64%
6	1	2	2	7,23%

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

Tratamiento	Factor			%GAP
	A	B	C	
7	1	3	3	0,00%
8	1	3	3	4,39%
9	1	3	3	0,04%
10	2	1	2	-8,21%
11	2	1	2	-14,08%
12	2	1	2	-5,78%
13	2	2	3	-18,88%
14	2	2	3	-4,51%
15	2	2	3	-1,34%
16	2	3	1	-65,26%
17	2	3	1	-2,63%
18	2	3	1	-0,12%
19	3	1	3	-38,23%
20	3	1	3	-66,16%
21	3	1	3	-51,06%
22	3	2	1	-75,11%
23	3	2	1	-59,95%
24	3	2	1	-58,56%
25	3	3	2	-52,42%
26	3	3	2	-77,44%
27	3	3	2	-81,62%

Para las instancias de 15 trabajos el algoritmo genético propuesto obtuvo en promedio resultados deficientes en 1,5% con respecto a la solución hallada por el B&C, mientras que para instancias con 30 y 60 trabajos se obtuvo en promedio soluciones mejores en 17% y 60%, respectivamente. Es importante notar que los resultados obtenidos para las instancias de 15 trabajos se logró con un tiempo de ejecución promedio de 0,4 minutos, mientras que para instancias de 30 trabajos el tiempo promedio es cercano a los 3 minutos. Finalmente, para las instancias de 60 trabajos se obtuvieron tiempos de ejecución de alrededor de 23 minutos. De esta forma, se aprecia que, aunque aumenta significativamente el tiempo de ejecución, se logra una mejora ostensible en la calidad de las soluciones a medida que aumenta la complejidad del problema. La calidad de las soluciones encontradas se

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

mantiene incluso si son comparadas con las obtenidas por medio del B&C en tiempos de ejecución más largos. Los resultados promedio se muestran en la Tabla 3.

Tabla 3. Resultados promedio por cantidad de trabajos

Trabajos	Promedio FO (GA)	Promedio FO (B&C)	%GAP	Tiempo Promedio (min)
15	5976,58	5898,05	1,33%	0,3792
30	11519,39	13929,02	-17,30%	2,9500
60	25245,17	76612,68	-67,05%	23,4898

Analizando el resultado del diseño de experimentos, se observa que los factores B y C no son relevantes en el desempeño del GA propuesto, mientras que el factor A (tamaño del problema) sí influye de manera significativa, tal y como se aprecia en la Figura 7.

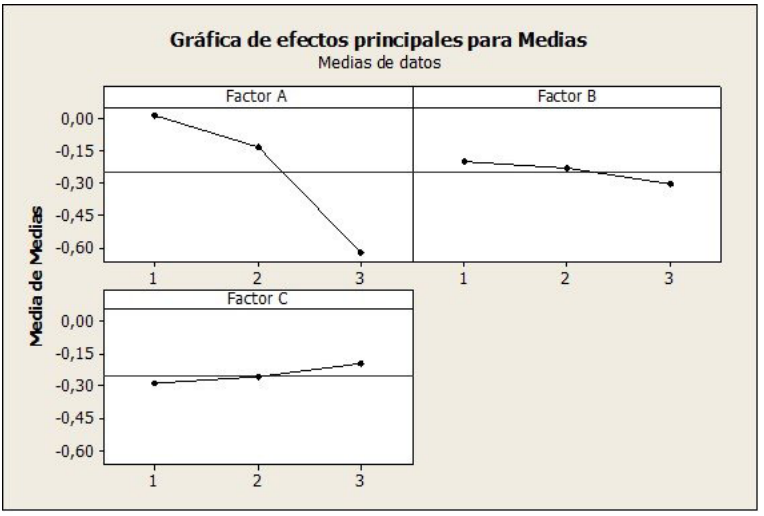

Figura 7. Efecto de los factores en la media del %GAP.

Con base en los resultados se observa que el método propuesto obtiene soluciones de costo total más bajo en comparación a las obtenidas con el método exacto en particular cuando el tamaño de las instancias aumenta. Sin embargo, el desempeño no varía significativamente si se modifican las condiciones operativas del problema en términos de los costos.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

CONCLUSIONES

En el presente artículo se desarrolla un algoritmo genético híbrido para la secuenciación de tamaños de lote con penalizaciones por terminaciones tempranas, tardías y costos de alistamiento dependientes de la secuencia. En el mismo se presenta una modificación a la heurística *Optimal Timming Algorithm* la cual se utiliza como mecanismo de cálculo para la función de adaptación.

En el caso de instancias de 15 trabajos el método propuesto logra soluciones en promedio 1,33% de costo total más alto que las halladas por el método de solución exacto, sin embargo esto lo hace en un tiempo promedio de 0,4 segundos lo cual es considerablemente inferior a las dos horas proporcionadas para el método exacto. Por el contrario, para instancias de 30 y 60 trabajos se obtienen soluciones en promedio 17,3% y 67,05% de costo total más bajo, respectivamente. Estas últimas se alcanzan en un tiempo computacional razonable.

El algoritmo propuesto obtiene soluciones robustas para un ambiente de manufactura de una sola máquina sin importar cuales sean las proporciones de los costos de alistamiento y penalización por tardanza con respecto al costo de mantener inventario.

En futuras investigaciones se podría implementar otras metaheurísticas o mecanismos de búsqueda que permitan mejorar el desempeño computacional y alcanzar resultados en menor tiempo para problemas con alto número de trabajos a programar.

REFERENCIAS

[1] M. Pinedo, *Scheduling. Theory, Algorithms, and Systems*, Third ed. Springer, 2008.

[2] R. Kuik, M. Salomon, y L. N. van Wassenhove, "Batching decisions: structure and models," *Eur. J. Oper. Res.*, vol. 75, no. 2, pp. 243–263, Jun. 1994. DOI:10.1016/0377-2217(94)90072-8

[3] W. Supithak, S. D. Liman, y E. J. Montes, "Lot-sizing and scheduling problem with earliness tardiness and setup penalties," *Comput. Ind. Eng.*, vol. 58, no. 3, pp. 363–372, Apr. 2010. DOI:10.1016/j.cie.2008.10.005

[4] A. Drexl, y A. Kimms, "Lot sizing and scheduling - survey and extensions," *Eur. J. Oper. Res.*, vol. 99, n.º 2, pp. 221–235, 1997. DOI:10.1016/S0377-2217(97)00030-1

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

[5] G. Belvaux and L. a. Wolsey, "Modelling Practical Lot-Sizing Problems as Mixed-Integer Programs," *Manage. Sci.*, vol. 47, no. 7, pp. 993–1007, 2001. DOI: 10.1287/mnsc.47.7.993.9800

[6] B. Karimi, S. M. T. Fatemi Ghomi, y J. M. Wilson, "The capacitated lot sizing problem: A review of models and algorithms," *Omega*, vol. 31, no. 5, pp. 365–378, 2003. DOI:10.1016/S0305-0483(03)00059-8

[7] P. Robinson, A. Narayanan, y F. Sahin, "Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms," *Omega*, vol. 37, no. 1, pp. 3–15, Feb. 2009. doi:10.1016/j.omega.2006.11.004

[8] A. Clark, B. Almada-Lobo, y C. Almeder, "Editorial: Lot sizing and scheduling: Industrial extensions and research opportunities," *Int. J. Prod. Res.*, vol. 49, no. 9, pp. 2458 - 2461, 2011. DOI:10.1080/00207543.2010.532908

[9] N. Absi, "Models and methods for capacitated lot-sizing problems," *4or*, vol. 6, no. 3, pp. 311–314, 2008.

[10] D. Gupta and T. Magnusson, "The capacitated lot-sizing and scheduling problem with sequence-dependent setup costs and setup times," *Comput. Oper. Res.*, vol. 32, no. 4, pp. 727–747, Apr. 2005. DOI:10.1016/j.cor.2003.08.014

[11] M. Caserta, y E. Q. Rico, "A cross entropy-Lagrangean hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times," *Comput. Oper. Res.*, vol. 36, no. 2, pp. 530–548, Feb. 2009. DOI:10.1016/j.cor.2007.10.014

[12] N. Absi, y S. Kedad-Sidhoum, "The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs," *Comput. Oper. Res.*, vol. 36, no. 11, pp. 2926–2936, Nov. 2009. doi:10.1016/j.cor.2009.01.007

[13] A. Kovács, K. N. Brown, y S. A. Tarim, "An efficient MIP model for the capacitated lot-sizing and scheduling problem with sequence-dependent setups," *Int. J. Prod. Econ.*, vol. 118, no. 1, pp. 282–291, Mar. 2009. DOI:10.1016/j.ijpe.2008.08.033

[14] I. S. Shim, H. C. Kim, H. H. Doh, y D. H. Lee, "A two-stage heuristic for single machine capacitated lot-sizing and scheduling with sequence-dependent setup costs," *Comput. Ind. Eng.*, vol. 61, no. 4, pp. 920–929, 2011. DOI:10.1016/j.cie.2011.06.002

[15] J. C. Lang, y Z. J. M. Shen, "Fix-and-optimize heuristics for capacitated lot-sizing with sequence-dependent setups and substitutions," *Eur. J. Oper. Res.*, vol. 214, no. 3, pp. 595–605, 2011. DOI:10.1016/j.ejor.2011.05.014

[16] B. Fleischmann, y H. Meyr, "The general lotsizing and scheduling problem," *OR Spektrum*, vol. 19, no. 1, pp. 11–21, 1997. DOI: 10.1007/BF01539800

[17] M. Salomon, y M. Solomon, "The discrete lotsizing and scheduling problem with sequence dependent set-up costs and set-up times using the travelling salesman problem with time windows," *Eur. J. Oper. Res.*, vol. 100, 1997.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

[18] K. Haase, y A. Kimms, "Lot sizing and scheduling with sequence-dependent setup costs and times and efficient rescheduling opportunities," *Int. J. Prod. Econ.*, vol. 66, pp. 159–169, 2000. DOI:10.1016/S0925-5273(99)00119-X

[19] P. França, A. Mendes, y P. Moscato, "A memetic algorithm for the total tardiness single machine scheduling problem," *Eur. J. Oper. Res.*, vol. 132, pp. 224–242, 2001. DOI:10.1016/S0377-2217(00)00140-5

[20] C. Gagné, W. Price, y M. Gravel, "Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times," *J. Oper. Res. Soc.*, vol. 53, no. 8, pp. 895–906, 2002.

[21] F. Sourd, "Earliness–tardiness scheduling with setup considerations," *Comput. Oper. Res.*, vol. 32, no. 7, pp. 1849–1865, Jul. 2005. DOI:10.1016/j.cor.2003.12.002

[22] J.-G. Kim, y D.-H. Lee, "Algorithms for common due-date assignment and sequencing on a single machine with sequence-dependent setup times," *J. Oper. Res. Soc.*, vol. 60, no. 9, pp. 1264–1272, Sep. 2008. DOI: 10.1057/jors.2008.95

[23] J. Montoya-Torres, M. Soto-Ferrari, F. González-Solano, and E. H. Alfonso-Lizarazo, "Machine scheduling with sequence-dependent setup times using a randomized search heuristic," in *Computers & Industrial Engineering, 2009. CIE 2009. International Conference on*, 2009, pp. 28–33. doi. 10.1109/ICCIE.2009.5223742

[24] N. Nekoiemehr, y G. Moslehi, "Minimizing the sum of maximum earliness and maximum tardiness in the single-machine scheduling problem with sequence-dependent setup time," *J. Oper. Res. Soc.*, vol. 62, no. 7, pp. 1403–1412, Jul. 2010.

[25] E. C. Sewell, J. J. Sauppe, D. R. Morrison, S. H. Jacobson, y G. K. Kao, "A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times," *J. Glob. Optim.*, vol. 54, no. 4, pp. 791–812, Oct. 2011. DOI: 10.1007/s10898-011-9793-z

[26] L. Guimarães, D. Klabjan, y B. Almada-Lobo, "Modeling lotsizing and scheduling problems with sequence dependent setups," *Eur. J. Oper. Res.*, vol. 239, pp. 644–662, 2013.

[27] S. A. Araújo, y A. Clark, "A priori reformulations for joint rolling-horizon scheduling of materials processing and lot-sizing problem," *Comput. Ind. Eng.*, vol. 65, no. 4, pp. 577–585, Aug. 2013. DOI:10.1016/j.cie.2013.04.003

[28] N. Sereshti, y M. Bijari, "Profit maximization in simultaneous lot-sizing and scheduling problem," *Appl. Math. Model.*, vol. 37, no. 23, pp. 9516–9523, Dec. 2013. DOI:10.1016/j.apm.2013.05.004

[29] Y. Yin, T. C. E. Cheng, C. J. Hsu, y C. C. Wu, "Single-machine batch delivery scheduling with an assignable common due window," *Omega*, vol. 41, no. 2, pp. 216–225, Apr. 2013. DOI:10.1016/j.omega.2012.06.002

[30] X. Li, H. Ishii, y T. Masuda, "Single machine batch scheduling problem with fuzzy batch size," *Comput. Ind. Eng.*, vol. 62, no. 3, pp. 688–692, 2012. DOI:10.1016/j.cie.2011.12.021

[31] Y. T. Hou, D. L. Yang, y W. H. Kuo, "Lot scheduling on a single machine," *Inf. Process. Lett.*, vol. 114, no. 12, pp. 718–722, 2014. DOI:10.1016/j.ipl.2014.06.016

[32] J. Pei, X. Liu, W. Fan, P. M. Pardalos, A. Migdalas, y S. Yang, "Scheduling jobs on a single serial-batching machine with dynamic job arrivals and multiple job types," *Ann. Math. Artif. Intell.*, 2015. DOI: 10.1007/s10472-015-9449-7

[33] C. Jordan y A. Drexl, "Discrete Lotsizing and Scheduling by Batch Sequencing," *Manage. Sci.*, vol. 44, no. 5, pp. 698–713, May 1998.

[34] C. Blum, J. Puchinger, G. R. Raidl, y A. Roli, "Hybrid metaheuristics in combinatorial optimization: A survey," *Appl. Soft Comput.*, vol. 11, no. 6, pp. 4135–4151, Sep. 2011. DOI:10.1016/j.asoc.2011.02.032

[35] E. G. Talbi, "Combining metaheuristics with mathematical programming, constraint programming and machine learning," *4or*, vol. 11, no. 2, pp. 101–150, Jul. 2013. DOI: 10.1007/s10288-013-0242-3

[36] L. Liu, y H. Zhou, "Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem," *Inf. Sci. (Ny).*, vol. 226, no. 37, pp. 68–92, Mar. 2013. DOI:10.1016/j.ins.2012.11.007

[37] S. W. Lin, y K. C. Ying, "A hybrid approach for single-machine tardiness problems with sequence-dependent setup times," *J. Oper. Res. Soc.*, vol. 59, no. 8, pp. 1109–1119, Jun. 2007.

[38] H. Xu, Z. Lü, A. Yin, L. Shen, y U. Buscher, "A study of hybrid evolutionary algorithms for single machine scheduling problem with sequence-dependent setup times," *Comput. Oper. Res.*, vol. 50, pp. 47–60, Oct. 2014. DOI:10.1016/j.cor.2014.04.009

[39] Z. Drezner, y A. Misevičius, "Enhancing the performance of hybrid genetic algorithms by differential improvement," *Comput. Oper. Res.*, vol. 40, no. 4, pp. 1038–1046, Apr. 2013. DOI:10.1016/j.cor.2012.10.014

[40] R. M'Hallah, "Minimizing total earliness and tardiness on a single machine using a hybrid heuristic," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3126–3142, Oct. 2007. DOI:10.1016/j.cor.2005.11.021

[41] C. Lee, y J. Choi, "A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights," *Comput. Oper. Res.*, vol. 22, no. 8, pp. 857–869, 1995. DOI:10.1016/0305-0548(94)00073-H

[42] M. Khouja, Z. Michalewicz, y M. Wilmot, "The use of genetic algorithms to solve the economic lot size scheduling problem," *Eur. J. Oper. Res.*, vol. 110, no. 3, pp. 509–524, Nov. 1998. DOI:10.1016/S0377-2217(97)00270-1

[43] J. Kim, "A hybrid genetic approach for single machine scheduling with distinct due dates and release times," in *Science and Technology, 1999. KORUS*

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

'99. Proceedings. The Third Russian-Korean International Symposium on, 1999, vol. 1, pp. 245-248. DOI. 10.1109/KORUS.1999.875916

[44] S. W. Lin, y K. C. Ying, "Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics," *Int. J. Adv. Manuf. Technol.*, vol. 34, no. 11–12, pp. 1183–1190, Aug. 2006. DOI: 10.1007/s00170-006-0693-1

[45] N. Tsavatapalli, y F. Badurdeen, "Single Machine Scheduling to Minimize Total Tardiness with Sequence-Dependent Setup Times," in *Proceedings of the 2009 Industrial Engineering Research Conference*, 2009, pp. 2122–2128.

[46] A. Husseinzadeh Kashan, B. Karimi, y F. Jolai, "An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes," *Eng. Appl. Artif. Intell.*, vol. 23, no. 6, pp. 911–922, Sep. 2010. DOI:10.1016/j.engappai.2010.01.031

[47] C. M. Joo and B. S. Kim, "Genetic algorithms for single machine scheduling with time-dependent deterioration and rate-modifying activities," *Expert Syst. Appl.*, vol. 40, no. 8, pp. 3036–3043, Jun. 2013. DOI:10.1016/j.eswa.2012.12.019

[48] V. Kodaganallur, A. K. Sen, y S. Mitra, "Application of graph search and genetic algorithms for the single machine scheduling problem with sequence-dependent setup times and quadratic penalty function of completion times," *Comput. Ind. Eng.*, vol. 67, pp. 10–19, Jan. 2014. DOI:10.1016/j.cie.2013.10.005

[49] F. H. Grupe, y S. Jooste, "Genetic algorithms: A business perspective," *Inf. Manag. Comput. Secur.*, vol. 12, no. 3, pp. 288–297, 2004. DOI: 10.1108/09685220410542624

[50] G. R. Raidl, J. Puchinger, y C. Blum, "Metaheuristic hybrids," in *Handbook of metaheuristics*, Second., M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, pp. 471–496.

[51] S. H. Chen, M. C. Chen, y Y. C. Liou, "Artificial chromosomes with genetic algorithm 2 (ACGA2) for single machine scheduling problems with sequence-dependent setup times," *Appl. Soft Comput. J.*, vol. 17, pp. 167–175, 2014. DOI:10.1016/j.asoc.2013.12.019

[52] G. Deng, y X. Gu, "An iterated greedy algorithm for the single-machine total weighted tardiness problem with sequence-dependent setup times," *Int. J. Syst. Sci.*, vol. 45, no. 3, pp. 351–362, 2012. DOI:10.1080/00207721.2012.723054

[53] H. Xu, Z. Lü, y T. C. E. Cheng, "Iterated local search for single-machine scheduling with sequence-dependent setup times to minimize total weighted tardiness," *J. Sched.*, vol. 17, no. 3, pp. 271–287, 2014. DOI: 10.1007/s10951-013-0351-z

[54] Q. Guo, y L. Tang, "An improved scatter search algorithm for the single machine total weighted tardiness scheduling problem with sequence-dependent setup times," *Appl. Soft Comput.*, vol. 29, pp. 184–195, 2015. doi:10.1016/j.asoc.2014.12.030

ANEXO

Formulación del BSPET como problema de Programación Entera Mixta

Los parámetros usados en el modelo son: B es un número muy grande, d_n es la fecha de entrega del trabajo n, h_n es el costo de mantener en el inventario una unidad del trabajo n, $J(n)$ es la función que identifica el tipo de producto al que pertenece el trabajo n, N es el total de trabajos a programar al inicio del horizonte de planeación, P_n es el tiempo de proceso del trabajo n, S_{ij} y CS_{ij} son respectivamente el tiempo y costo de alistamiento por pasar del trabajo i al trabajo j respectivamente y T_{cn} es el costo unitario por terminación tardía del trabajo n. Las variables de decisión del problema son: r_n y s_n son los tiempos de terminación e inicio del trabajo n respectivamente, $x_{nk}P_n$ es una variable binaria que toma valor de 1 si el trabajo n es programado antes que el trabajo k, t_n y e_n son la tardanza y entrega temprana del trabajo n respectivamente.

La formulación del BSPET se muestra a continuación:

Función objetivo:

$$Min\ Z = \sum_{n=1}^{N}\sum_{k=1}^{N} CS_{J(n)J(k)}\, x_{nk} + \sum_{n=1}^{N} h_{J(n)}\, e_n + \sum_{k=1}^{N} TC_{J(n)}\, t_n \quad (11)$$

Sujeto a las siguientes restricciones:

$$\sum_{k=1}^{N} x_{nk} = 1 \qquad n = 1,\ldots,N, \quad n \neq k \quad (12)$$

$$\sum_{k=1}^{N} x_{nk} = 1 \qquad k = 1,\ldots,N, \quad n \neq k \quad (13)$$

$$s_n + p_n + S_{J(n)\,J(k)} \leq s_k + B(1 - x_{nk}) \qquad \begin{matrix} n = 1,\ldots,N \\ k = 1,\ldots,N \\ n \neq k \end{matrix} \quad (14)$$

$$r_n = s_n + p_n \qquad n = 1,\ldots,N \quad (15)$$

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)

$$r_n - d_n \leq t_n \qquad n = 1,\ldots,N \qquad (16)$$

$$d_n - r_n \leq e_n \qquad n = 1,\ldots,N \qquad (17)$$

$$r_n,\ s_n,\ t_n,\ e_n \geq 0 \qquad n = 1,\ldots,N$$
$$x_{nk} \in \{0,1\} \qquad k = 1,\ldots,N \qquad (18)$$

La función objetivo (11) es la minimización de la sumatoria de los costos de alistamiento, de mantener unidades en el inventario y de penalizaciones por entregas tardías de los trabajos. Las restricciones (12) y (13) garantizan que cada trabajo tenga un solo sucesor y un solo predecesor, respectivamente. La restricción (14) realiza el cálculo de los tiempos de inicio de los trabajos, al igual que la definición de las variables binarias , las cuales determinan la secuencia de los trabajos. La restricción (15) realiza el cálculo de los tiempos de terminación de los trabajos, mientras que las restricciones (16) y (17) realizan el cálculo de las tardanzas y de las terminaciones tempranas de los mismos, respectivamente. (18) son las restricciones lógicas del problema.

ISSN: 0122-3461 (*impreso*)
2145-9371 (*on line*)