ISSN 1870-4069

Propuesta de algoritmo GRASP para el problema de la tardanza total ponderada en una máquina

Rosmery-Caridad Acosta-Núñez, Guillermo-Airam Piloto-Cuellar, Ricardo Hernández-Zitlalpopoca, Santiago-Omar Caballero-Morales

UPAEP, Puebla, México
{rosmerycaridad.acosta, guillermoairam.piloto, ricardo.hernandez02, santiagoomar.caballero}@upaep.mx

Resumen. En este artículo se aborda el problema de la tardanza total ponderada en una máquina, conocido como Single Machine Total Weighted Tardiness (SMTWT, por sus siglas en inglés), reconocido en la literatura como un problema de tipo NP-duro debido a su complejidad computacional. Para su solución se hace una propuesta de algoritmo Greedy Randomized Adaptive Search Procedure (GRASP) con una estrategia de integración de los operadores de diversificación 2-opt e inversión, cuyo objetivo es la minimización de costos a través de encontrar un orden efectivo de las tareas abordadas en cada máquina. Se utilizó la biblioteca OR-Library con problemas de 40, 50 y 100 tareas con 125 instancias cada uno. El algoritmo propuesto obtuvo, para el total de casos analizados, valores que en promedio tenían un 6.5% de error respecto a los mejores valores conocidos y en aproximadamente el 15% se encontraron mejores soluciones que aquellas reportadas en la literatura.

Palabras clave: búsqueda local, GRASP, inversión, intercambio, SMTWT.

Grasp Algorithm Proposed for the Problem of the Single Machine Total Weighted Tardiness

Abstract. This article addresses the problem of total weighted delay in a machine, known as Single Machine Total Weighted Tardiness (SMTWT) for its acronym in English, recognized in the literature as a NP-hard problem due to its computational complexity. For its solution, a Greedy Randomized Adaptive Search Procedure (GRASP) algorithm is proposed with an integration strategy of 2-opt and inversion diversification operators, whose objective is to minimize costs through finding an effective order of tasks addressed in each machine. We used the OR-Library library with problems of 40, 50 and 100 tasks with 125 instances each. The proposed algorithm obtained for the total of analyzed cases values that on average had a 6.5% error with respect to the best known values and in approximately 15% of the cases better solutions were found than those reported in the literature.

Keywords: local search, GRASP, SMTWT, inversion, 2-Opt.

1. Introducción

La programación de trabajos comenzó a tomarse en serio en los procesos de producción a principios de este siglo con el trabajo de Henry Gantt y otros pioneros. Consiste en un proceso de toma de decisiones que se utiliza de forma regular en muchas industrias de producción y servicios, sobre el principio de la asignación de recursos a tareas en períodos de tiempo determinados y su propósito es optimizar uno o más objetivos, como mínimo tiempo, mayor beneficio, mayor utilización, entre otros. La programación, como proceso de toma de decisiones, juega un papel importante en la mayoría de los sistemas de fabricación y producción, así como en la mayoría de los entornos de procesamiento de la información. También es importante en entornos de transporte y distribución y en otros tipos de industrias de servicios [1].

Según Davoudpour [2], la planificación o diseño de la producción es un problema de decisiones a lo largo de la cadena de suministro. Esta planificación en la logística, implica la asignación de tareas o servicios para optimizar procesos en un área manufacturera.

La agrupación de estas tareas es necesaria dadas ciertas cuestiones tecnológicas y procesos de capacidad: una vez que se tenga la asignación de una familia de productos por máquina, se puede gestionar la asignación de tareas para cada una de las máquinas y obtener un sistema de producción homogenizado [3]. En otras palabras, el objetivo es completar una serie de trabajos en una sola máquina tan cercana como lo sea la fecha de entrega del cliente, la minimización de la tardanza es un concepto de suma importancia en la industria y sistemas de manufactura esbelta conocido como justo a tiempo, Just-in-Time (JIT, por sus siglas en inglés) [4].

En este artículo se aborda el problema de ordenamiento de tareas en una sola máquina, con el objetivo de minimizar la tardanza total ponderada. Para esto se analizaron tres tipos de problemas de secuenciación: de 40, 50 y 100 trabajos. Cada tipo de problema está compuesto de 125 instancias (diferentes ejemplos de problemas con el número de trabajos especificado) y fueron obtenidos de la biblioteca digital OR-Library. Con el fin de encontrar la secuencia que minimice costos, se implementó el algoritmo de búsqueda local GRASP con una estrategia de aplicación de los operadores 2-opt e inversión.

Los avances del artículo se presentan de la siguiente manera: en la Sección 2 se presenta una revisión bibliográfica sobre los antecedentes del problema abordado; en la Sección 3 se desarrolla un análisis del método de resolución aplicado, sucedida por la Sección 4, donde se muestran los resultados obtenidos; finalmente, en la Sección 5 se presentan las conclusiones referentes al análisis de los aportes de la investigación.

2. Estado de arte

El problema de una máquina ha sido el más popular de todos los problemas de programación. Existe una amplia variedad de resultados para un amplio conjunto de especificaciones de problemas. Dentro de estos resultados se encuentran los procedimientos para minimizar el tiempo medio de flujo [5] y para minimizar la

tardanza máxima [6]; ambos procedimientos determinan la secuencia de tareas óptima por medio de un orden simple de las tareas. Un procedimiento un poco más complejo es el de Moore [7], para minimizar el número de tareas tardías; aquí las tareas se ordenan, primero, de acuerdo con los tiempos de finalización deseados y, luego, esta secuencia se modifica mediante la eliminación secuencial de tareas tardías. Si bien los problemas antes mencionados han demostrado ser bastante fáciles, el problema de minimizar la tardanza ponderada ha sido considerablemente más difícil [8].

Pinedo [1] expone varias razones de por qué los modelos de una sola máquina son importantes. Estos modelos a menudo tienen propiedades que ni las máquinas en paralelo ni las máquinas en serie tienen, por lo cual, los resultados que se pueden obtener no sólo proporcionan información sobre su entorno, sino que, además, suministran una base para la heurística que se aplica a entornos de máquinas más complicados. En la práctica, los problemas de programación para estos entornos a menudo se descomponen en sub-problemas que se ocupan de máquinas individuales.

La programación de producción se puede definir como la asignación de recursos de producción disponibles a lo largo del tiempo para satisfacer mejor un conjunto de criterios. Típicamente, el problema de programación implica un conjunto de tareas a realizar, y los criterios pueden involucrar compensaciones entre la finalización temprana y tardía de una tarea, y mantener el inventario para la tarea y frecuentes cambios de producción [8].

Diversos estudios han abordado el problema de la tardanza. Karp [9] demostró que este problema es NP-completo. Srinivasan [10] determinó relaciones de precedencia entre pares de trabajos, lo cual planteó una serie de proposiciones que redujeron considerablemente el cálculo total involucrado en la fase de programación dinámica del método de solución. Igualmente, Fisher [11] presentó un algoritmo Branch & Bound para el problema de la programación de trabajos en una sola máquina para minimizar las tardanzas. El principal resultado teórico de su trabajo fue un algoritmo que resuelve el problema de Lagrange en una serie de pasos proporcionales al producto de *n2* y el tiempo promedio de procesamiento del trabajo.

Asimismo, Rinnooy Kan *et al.* [12], Picard y Queyranne [13] y Baker y Schrage [14] estudiaron el problema de la tardanza de una máquina o una versión generalizada de la misma. Schrage y Baker [15] implementaron un algoritmo de programación dinámica para resolver problemas de secuenciación de una máquina con restricciones de precedencia. Dicho algoritmo fue mucho más eficiente que los anteriores para ciertos problemas de secuenciación de una máquina. El trabajo de Graves [8] ofrece un grupo de observaciones sobre el estado de la práctica de la programación de producción, estableciendo una perspectiva con la cual contrasta la teoría de programación.

El Single Machine Total Weighted Tardiness o SMTWT, que se conoce como $1||\sum w_j T_j$ según Graham *et al.* [16], se puede identificar como NP-duro. Lo que significa que se requiere de un alto esfuerzo computacional para ejecutar un algoritmo de optimización combinatoria, mientras que otros problemas, como la programación dinámica o la ramificación, tienen una complejidad cuadrática [21].

Bozejko, Grabowski y Wodecki [18] también han abordado el problema de la tardanza ponderada, donde proponen un algoritmo de búsqueda tabú con un

vecindario específico que emplea bloques de trabajos y una técnica de movimientos compuestos a partir de algunas propiedades nuevas del problema asociado con los bloques.

Otra revisión donde se aborda la tardanza de una sola máquina a partir de un algoritmo de búsqueda tabú junto a otros meta-heurísticos es Liao y Cheng [19]. Asimismo, Navarrete *et al.* [20], Caballeros y Alvarado [21], Vega y Caballeros [22], abordan la minimización de la tardanza total ponderada en un entorno de producción mediante el meta-heurístico de Procedimientos de Búsqueda Voraces Aleatorizados y Adaptativos o Greedy Randomize Adaptive Search Procedure (GRASP, por sus siglas en ingles). Este meta-heurístico es constructivo y determinístico dado que genera una solución completa calculando prioridades para cada uno de los trabajos a ser asignados.

Al añadir aleatoriedad en la selección de las tareas para su asignación se da la oportunidad de asignar trabajos diferentes a aquellos determinados mediante el proceso determinístico, lo cual puede conllevar a una mejor asignación global [23].

Otras de las posibles metodologías estudiadas para la solución del problema de la tardanza total ponderada en una máquina son, el procedimiento de la búsqueda vecina cercana [24], la secuencia inicial en dos pasos [24], el algoritmo de recocido simulado [25], algoritmos voraces [26], y algoritmos genéticos [27].

Una vez seleccionado el método de solución a utilizar en un problema de optimización, vale la pena comprobar su comportamiento en diferentes escenarios. La prueba *t* de dos muestras se puede utilizar para comparar los promedios entre grupos y determinar si existe alguna diferencia estadística significativa entre ellos. Esto para descartar diferencias causadas por eventos aleatorios y no por la eficiencia de un método con respecto a otro.

La prueba de hipótesis, es una prueba de análisis multi-variable en dos vectores de medias μ_1 y μ_2, basado en dos muestras aleatorias independientes, una con distribución de medias μ_1 y otra con una distribución de medias μ_2. Esta prueba es utilizada en diferentes aplicaciones científicas [28]. La clásica formulación de hipótesis se expresa como:

$$\text{Hipótesis Nula} \quad \text{Ho: } \mu_1 - \mu_2 = 0,$$
$$\text{Hipótesis Alternativa} \quad \text{Ha: } \mu_1 - \mu_2 \neq 0.$$

Para determinar si la diferencia entre las muestras de medias es estadísticamente significativa, es posible utilizar la comparativa de valor *p* con respecto al nivel de significancia α que nos indica el nivel de riesgo que existe en que realmente no haya diferencia. Es posible hacer las siguientes aseveraciones para el análisis de las muestras en una prueba de hipótesis [28]:

- $p \leq \alpha$: La diferencia entre las medias es estadísticamente significativa – Si *p* es menor o igual que el nivel de significancia, se rechaza la hipótesis nula, y se puede concluir que existen diferencias entre las medias de la población.
- $p > \alpha$: La diferencia entre las medias no es estadísticamente significativa – Si *p* es mayor al nivel de significancia, la decisión es no rechazar la hipótesis nula, pues no hay evidencia suficiente para concluir que la diferencia entre las medias de las muestras, sean estadísticamente significantes.

3. Metodología aplicada

Búsqueda local es uno de los algoritmos más utilizados en el problema SMTWT encontrados en la literatura. A continuación, se explica en qué consiste el problema de SMTWT, el algoritmo GRASP en su forma estándar y el seudocódigo del GRASP propuesto en el presente artículo.

3.1. SMTWT

Según Abdul-Razaq *et al.* [29] el problema de SMTWT puede expresarse formalmente de la siguiente forma: cada uno de los *n* trabajos (numerados 1, ..., *n*) debe procesarse sin interrupción en una sola máquina que pueda manejar sólo un trabajo a la vez. Para un mejor entendimiento del problema, se expone un ejemplo dónde *n*=6, en la Tabla 1 se muestra la secuencia inicial de las tareas para el análisis.

Tabla 1. Secuencia de tareas para una máquina.

1	2	3	4	5	6

El trabajo *i* (*i* = 1, …, *n*) queda disponible para el procesamiento en el tiempo cero. Esto significa que todas las tareas pueden ser programadas para comenzar la secuencia y por lo tanto no hay dependencia entre ellas. Cada tarea requiere un tiempo de procesamiento entero p_i, un peso positivo w_i y una fecha de vencimiento o de entrega d_i. Estos datos se presentan en la Tabla 2 para el análisis del ejemplo.

Tabla 2. Datos para el ejemplo analizado.

Tarea (*i*)	p_i	w_i	d_i
1	5	1	5
2	6	4	10
3	8	3	15
4	4	4	9
5	7	2	20
6	9	1	30

Para un orden de procesamiento de los trabajos, para el ejemplo se utilizará la secuencia expuesta en la Tabla 1, el tiempo de finalización (más temprano) se puede calcular como $C_i = P_i + C_{i-1}$ y la tardanza $T_i = max\{C_i - d_i, 0\}$ de la tarea $i(i = 1, \dots, n)$. En la Tabla 3 se muestran los resultados para C_i y T_i.

Tabla 3. Tiempo de finalización más temprano y tardanza del ejemplo.

Tarea (*i*)	p_i	C_i	d_i	T_i
1	5	5	5	0
2	6	11	10	1
3	8	19	15	4
4	4	23	9	14
5	7	30	20	10
6	9	39	30	9

El objetivo de este problema es encontrar una secuencia de programación de las tareas que minimize la tardanza total $\sum_{i=1}^{n} w_i * T_i$, que en este ejemplo sería de 101 unidades, en la Tabla 4 se muestran los cálculos realizados para determinar dicho valor.

Tabla 4. Cálculo de la tardanza total ponderada para el ejemplo.

Tarea (i)	w_i	T_i	$w_i \times T_i$
1	1	0	0
2	4	1	4
3	3	4	12
4	4	14	56
5	2	10	20
6	1	9	9
$\sum_{i=1}^{n=6} w_i \times T_i$			101

Como se observa en la Tabla 4 para el ejemplo se tienen cinco tareas con tardanza, representando el 84% del total de las tareas a programar, por lo que es necesario encontrar otra secuencia de tareas que minimice la cantidad de trabajos con entrega tardía y a la vez la tardanza total ponderada.

3.2. GRASP

GRASP es una meta-heurística de búsqueda local de inicio múltiple, conformada por dos fases: a) en la primera se desarrolla un algoritmo GREEDY para encontrar aleatoriamente una solución no óptima, pero si factible al problema; b) en la segunda fase de realiza una búsqueda local y se comparan los beneficios de las nuevas soluciones encontradas con la solución histórica, y en caso de que sea mejor que esta última se procede a actualizar la histórica. A continuación, se observa el seudocódigo de un procedimiento general del GRASP propuesto por Resende y Ribeiro [30].

Procedure GRASP;

```
1     f* ← ∞;
2     for k = 1, …, Max.Iterations do
3           Construct a greedy randomized x ∈ X;
4           Find y by applying local search to x;
5           if f(y) < f* do;
6             x* ← y;
7             f* ← f( x*);
8           end if
9     end for
10    return x*
```

end GRASP;

3.3. GRASP propuesto

En el artículo se evalúa un conjunto de instancias mediante el meta-heurístico GRASP, en el cual se integraron de manera estratégica los heurísticos 2-opt e inversión. A continuación, se presenta el seudocódigo del algoritmo desarrollado para la solución de los problemas de SMTWT abordados:

$Procedure\ GRASP\ modified;$

1 $Read\ Input;$
2 $x^* \leftarrow Construct\ a\ greedy\ randomized\ x \in X;$
3 $f^* \leftarrow f(x^*);$
4 $for\ k = 1, \dots, Max.Iterations\ do$
5 $Construct\ a\ 2-opt\ randomized\ y \in Y;$
6 $Construct\ an\ reverse\ randomized\ z \in Z;$
7 $Find\ S\ by\ applying\ local\ search\ to\ y\ and\ z;$
8 $if\ f(S) < f^*\ do;$
9 $x^* \leftarrow S;$
10 $f^* \leftarrow f(x^*);$
11 $end\ if$
12 $end\ for$
13 $return\ x^*$

$end\ GRASP\ modified;$

Operadores de diversificación 2-opt e inversión. El operador de intercambio o 2-opt es un algoritmo iterativo de mejora. Este procede bajo una prueba sistemática, dependiendo si la secuencia actual puede ser mejorada intercambiando 2 tareas; para el 2-opt se requiere un tiempo de O(n2), por lo que se vuelve muy complejo cuando el número de tareas crece, por lo que el esfuerzo computacional es muy alto. En la Figura 1 se muestra cómo funciona el 2-opt tomando el ejemplo anterior para el problema de SMTWT en donde se seleccionan aleatoriamente dos posiciones.

a) b)

Fig. 1. Aplicación del 2-opt: a) Secuencia actual, b) Secuencia resultante.

Adicionalmente, se utilizó el operador de inversión, el cual es un algoritmo de optimización discreto continuo para coordinar los valores de tareas o nodos asignados en un problema, se toma un número de tareas, y se invierten de adelante hacia atrás, con el objetivo de encontrar un resultado mejor al anterior [31]. En la Figura 2 se muestra cómo funciona el operador de inversión tomando el ejemplo anterior para el problema de SMTWT en donde se seleccionan aleatoriamente dos posiciones para determinar el rango de tareas a las que se va a aplicar dicho operador. En este caso las

posiciones seleccionadas son 2 y 5 por lo que se realiza la inversión de las tareas comprendidas entre estas dos posiciones.

a) b)

Fig. 2. Aplicación del operador inversión: a) Secuencia actual, b) Secuencia resultante

En el algoritmo diseñado en el presente artículo se utilizan estos dos operadores para realizar la generación de nuevas soluciones a partir de una solución inicial, con la diferencia de que, al aplicar el operador de inversión a una serie de tareas, estas se extraen de la secuencia actual, se invierten y se insertan al final de la cadena no en el lugar dónde se encontraba inicialmente, para el ejemplo analizado anteriormente quedaría como se muestra en la Figura 3.

a) b)

Fig. 3. Aplicación del operador inversión con inserción: a) Secuencia actual, b) Secuencia resultante

Estos dos heurísticos se gestionaron a través del algoritmo GRASP, dónde en cada iteración se ejecutan ambos operadores y se selecciona la solución que mejor costo arroje una vez evaluada la función objetivo y en caso de que la seleccionada sea mejor que la histórica, se procede a su actualización, repitiendo dicho proceso hasta el máximo de iteraciones definidas.

4. Análisis de resultados

Se aplicó el algoritmo GRASP desarrollado en la sección anterior para la solución del problema SMTWT utilizando las instancias de J. E. Beasley de 40, 50 y 100 trabajos, conformado cada uno por 125 instancias. Estas se obtuvieron a través de la base de datos OR Library [32] la cual contiene un amplio número de problemas de investigación de operaciones que sirven de referencia para la comparación de algoritmos y técnicas de optimización a nivel mundial. Los datos para los problemas seleccionados se generaron aleatoriamente basándose en una distribución uniforme con valores entre 1 y 100 para los tiempos de procesamiento p_i, entre 1 y 10 para los pesos w_i y con una combinación de dos distribuciones uniformes para las fechas de entrega d_i. Dicha biblioteca cuenta actualmente con las soluciones óptimas para 124 de 125 instancias de 40 trabajos y 115 de 125 para 50 trabajos, así como las mejores soluciones encontradas para la instancia 19 de 40 trabajos, para los 10 restantes de 50 trabajos y para las 125 instancias de 100 trabajos.

Para la implementación de la meta-heurística seleccionada se trabajó en el software Matlab versión 2015a. Una vez obtenidos los resultados se compararon con los

óptimos y mejores conocidos reportados en la biblioteca, calculándose el error por cada problema, y se determinó que los mejores resultados fueron obtenidos mediante la combinación de los operadores de inversión aleatoria con inserción e intercambio aleatorio, además el tiempo de ejecución computacional para los tres tipos de problema fue en promedio de 33.39 minutos.

En las Figs. 4, 5 y 6 se observa la desviación porcentual de los valores obtenidos respecto a los valores óptimos o mejores conocidos de las tardanzas ponderadas para las 125 instancias en cada tipo de problema.

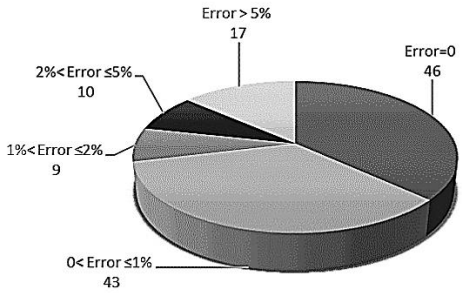

Fig. 4. Error obtenido por el algoritmo GRASP para las 125 instancias de 40 trabajos.

Como se observa en la Figura 4 el algoritmo desarrollado encontró los valores óptimos en 46 instancias que representan el 36.8% de las 125 analizadas, y en el 86.4% la desviación con respecto a los mejores valores reportados en la librería es menor al 5%. Por ende, de las 125 instancias de 40 trabajos analizadas inicialmente sólo el 13.6% tuvo una desviación superior al 5%. Siendo la desviación promedio de las 125 instancias de un 3.07% aproximadamente.

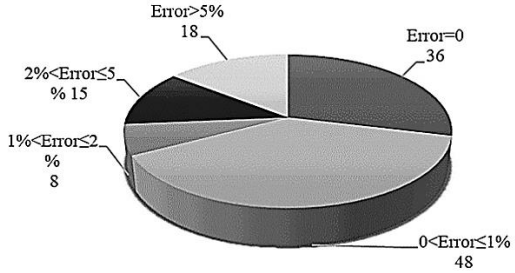

Fig. 5. Error obtenido por el algoritmo GRASP para las 125 instancias de 50 trabajos.

En la Fig. 5 se muestra que para las 125 instancias de 50 trabajos el método GRASP encontró 36 valores óptimos que representan el 28.8% de las 125 analizadas y que el 85.6% de los resultados obtenidos tienen una desviación menor al 5%, siendo la desviación promedio de las 125 instancias de un 2.84% aproximadamente.

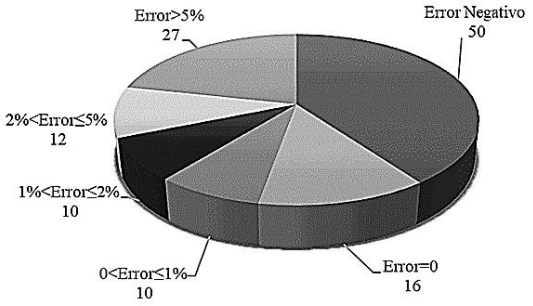

Fig. 6. Error obtenido por el algoritmo GRASP para las 125 instancias de 100 trabajos.

En el caso de las instancias de 100 trabajos no se tiene referencia de los valores óptimos, por lo que la desviación se corresponde con los mejores valores reportados hasta la actualidad en la base de datos. como se muestra en la Figura 6 de las 125 instancias analizadas con el método GRASP se encontró 50 soluciones para las cuales la evaluación en la función objetivo es mejor que el mejor conocido, representando un 40% de las 125 analizadas. Además, un 12.8% se corresponde con los mejores valores conocidos; y el 78.4% de los resultados obtenidos tienen una desviación menor al 5%, siendo la desviación promedio de las 125 instancias de un 13.74% aproximadamente.

Dados los resultados obtenidos, se trabajó con un comparativo de errores para identificar si éstos se comportan de forma similar. Para ello, se separaron los errores en tres poblaciones diferentes, errores de los problemas de 40, 50 y 100 tareas; el objetivo es aceptar o rechazar la idea que el meta-heurístico tiene la misma eficiencia en el total de instancias. De esta manera, con base a los datos de la población de los errores, se trabajará inicialmente con una prueba de hipótesis de igualdad y diferencia, analizando los datos opuestos sobre una población, una hipótesis nula y una hipótesis alternativa, la primera verificará si los datos de la población son iguales, y la alterna si los datos de la población son diferentes, el estadístico utilizado es la prueba "*t*" en un análisis de dos medias, aplicado para comparar si el promedio de dos poblaciones son significativamente diferentes. En base a ello, determinaremos la respuesta utilizando *p value* y se aceptará o rechazara la hipótesis con respecto al nivel de significancia: si *p value* es menor que el nivel de significancia α = 95% entonces se rechazará la hipótesis nula. A continuación, se muestran las poblaciones utilizadas para el análisis de significancia estadística:

- Errores de las instancias de 40 tareas = μ_1,
- Errores de las instancias de 50 tareas = μ_2,
- Errores de las instancias de 100 tareas = μ_3.

4.1. Pruebas de Significancia Estadística: Comparación de Medias

Primera prueba:

Ho: $\mu_1 - \mu_2 = 0$,

Ho: $\mu_1 - \mu_2 \neq 0$.

Resultados estadísticos de la primera prueba de hipótesis (resultados obtenidos mediante Minitab).

```
Two-Sample T-Test and CI: 40 tareas, 50 tareas

Two-sample T for 40 tareas vs 50 tareas

N    Mean   StDev  SE Mean
40 tareas  125  0.0307  0.0858   0.0077
50 tareas  125  0.0284  0.0700   0.0063

Difference = μ (40 tareas) - μ (50 tareas)
Estimate for difference:  0.00238
95% CI for difference: (-0.01713, 0.02188)
T-Test of difference = 0 (vs ≠): T-Value = 0.24 P-Value = 0.810
DF = 238
```

Segunda prueba:

Ho: $\mu_1 - \mu_3 = 0$,

Ho: $\mu_1 - \mu_3 \neq 0$.

Resultados estadísticos de la segunda prueba de hipótesis (resultados obtenidos mediante Minitab).

```
Two-Sample T-Test and CI: 40 tareas, 100 tareas

Two-sample T for 40 tareas vs 100 tareas

              N    Mean   StDev  SE Mean
40 tareas   125  0.0307  0.0858   0.0077
100 tareas  125   0.137   0.826    0.074

Difference = μ (40 tareas) - μ (100 tareas)
Estimate for difference:  -0.1067
95% CI for difference: (-0.2536, 0.0403)
T-Test of difference = 0 (vs ≠): T-Value = -1.44 P-Value =0.153
DF = 126
```

En ambos casos, podemos observar que el *p-value* es mayor al nivel de significancia, por lo que estadísticamente podemos asegurar que los errores se comportan de manera similar entre los diferentes problemas: 40, 50 y 100 tareas, con mucha más similitud entre las instancias de 40 vs. 50 tareas, y con una menor similitud entre las instancias de 40 vs 100. De esta manera se puede concluir que no hay evidencia suficiente para afirmar que existe una diferencia en el comportamiento de los resultados obtenidos por el algoritmo propuesto para los diferentes tipos de problema. Por lo tanto, el desempeño del algoritmo GRASP propuesto es consistente y robusto a través de problemas de asignaciones de diferentes tamaños.

5. Conclusiones y trabajo futuro

En este artículo se trabajó con una técnica de rápida implementación para encontrar una buena solución al problema de SMTWT. Considerando que este tipo de problemas se le considera NP-duro, se utilizó la integración de un meta-heurístico GRASP con los operadores 2-opt e inversión, y se compararon los resultados con los mejores valores conocidos de las diferentes instancias de los problemas. Este análisis y desarrollo de la solución presenta una alternativa competitiva y ofrece resultados rápidos para la secuenciación de tareas en máquinas de producción en línea. La principal contribución del trabajo se puede ver como el beneficio de sólo integrar dos heurísticos (2-opt e inversión) en un proceso de selección cerrada.

La calidad de los resultados es competitiva, se trabajó con 125 instancias para cada tipo de problema de 40, 50 y 100 tareas, haciendo un total de 375 instancias. En promedio, se obtuvo resultados con un porcentaje de error menor al 5% para alrededor del 83.47% (313 de 375). El algoritmo propuesto reportó resultados eficientes dado el error promedio de 6.5%, y, además, para 50 instancias de 100 tareas reportó mejores resultados que los conocidos.

El tiempo de procesamiento del algoritmo fue corto (33.39 minutos en promedio), en el cual se encontraron estos resultados, por lo que es posible encontrar valores óptimos y cercanos al óptimo en un tiempo razonable. Es importante mencionar que el tiempo de ejecución para los 3 problemas; 40, 50 y 100 tareas no es lineal.

El comportamiento del meta-heurístico desarrollado es adecuado. Como se pudo observar en el estadístico de prueba, no hay evidencia suficiente para determinar que los errores tienen una dispersión no-consistente en los problemas de 40, 50 y 100 tarea. Si bien, *p-value* no identificó comportamiento de dispersión, si se pudo observar que en el problema de 100 tareas la variación en los errores fue mayor, unos por arriba de lo esperado y otros mejor que los actuales conocidos.

Para trabajos futuros se sugiere comprobar con otros meta-heurísticos y operadores de diversificación, pues no se descarta la posibilidad de poder encontrar mejores resultados que los registrados por el algoritmo de búsqueda local propuesto para el problema SMTWT.

Referencias

1. Pinedo, M. L.: Scheduling: theory, algorithms, and systems. 3rd ed. Springer, NY (2008)
2. Davoudpour, G. R.: Coordinated Location, Distribution and Inventory Decisions in Supply Chain. Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran (2015)
3. Schaller, J.: Scheduling a permutation flow shop with family setups to minimize total tardiness. International Journal of Production Research 50(8), pp. 2204–2217 (2012)
4. Rabadi, G., Moraga, R., Al-Salem, A.: Heuristics for the unrelated parallel machine scheduling problem with setup times. Journal of Intelligent Manufacturing 17(1), pp. 85–97 (2006)
5. Smith, W. E.: Various optimizers for single stage production. Naval Research Logistics (NRL) 3(1-2), pp. 59–66 (1956)

6. Jackson, J. R.: Scheduling a Production Line to Minimize Maximum Tardiness. Research Report 43, Management Sciences Research Project, University of California, Los Angeles (1955)
7. Moore, J. M.: An n-Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs. Management Science 15(1), pp. 102–109 (1968)
8. Graves, S. C.: A review of production scheduling. Operations research 29(4), pp. 646–675 (1981)
9. Karp, R. M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series, pp 85–103. Springer, Boston, MA (1972)
10. Srinivasan, V.: A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. Naval Research Logistics (NRL) 18(3), pp. 317–327 (1971)
11. Fisher, M. L.: A dual algorithm for the one-machine scheduling problem. Mathematical programming 11(1), pp. 229–251 (1976)
12. Rinnooy Kan, A. H. G., Lageweg, B. J., Lenstra, J. K.: Minimizing total costs in one-machine scheduling. Operations Research 23(5), pp. 908–927 (1975)
13. Picard, J. C., Queyranne, M.: The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Operations Research 26(1), pp. 86–110 (1978)
14. Baker, K. R., Schrage, L. E.: Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks. Operations Research 26(1), pp. 111–120 (1978)
15. Schrage, L., Baker, K. R.: Dynamic programming solution of sequencing problems with precedence constraints. Operations research 26(3), pp. 444–449 (1978)
16. Graham, R. L., Lawler, E. L., Lenstra, J. K., Kan, A. R.: Optimization and approximation in deterministic sequencing and scheduling: a survey. In: P.L., Hammer, E.L. Johnson, B.H. Korte, (eds) Annals of Discrete Mathematics, vol. 5, pp. 287–326. Elsevier (1979)
17. Bożejko, W., Grabowski, J., Wodecki, M.: Block approach tabu search algorithm for single machine total weighted tardiness problem. Computers & Industrial Engineering 50(1–2), pp. 1–14 (2006)
18. Liao, C. J., Cheng, C. C.: A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. Computers & Industrial Engineering 52(4), pp. 404–413 (2007)
19. Niño-Navarrete, Á. M., Caballero-Villalobos, J. P.: Evaluación de funciones de utilidad de GRASP en la programación de producción para minimizar la tardanza total ponderada en una máquina. Ingeniería 14(2), pp. 51–58 (2009)
20. Caballero-Villalobos, J.P., Alvarado-Valencia, J.A.: Greedy Randomized Adaptive Search Procedure (GRASP), una alternativa valiosa en la minimización de la tardanza total ponderada en una máquina. Ingeniería y Universidad 14(2), 275–295 (2010)
21. Vega-Mejía, C. A., Caballero-Villalobos, J. P.: Uso combinado de GRASP y Path-Relinking en la programación de producción para minimizar la tardanza total ponderada en una máquina. Ingeniería y Universidad 14(1), 79–96 (2010)
22. Moreira, J. M.: Greedy randomized dispatching heuristics for the single machine scheduling problem with quadratic earliness and tardiness penalties. The International Journal of Advanced Manufacturing Technology 44(9–10), pp. 995–1009 (2009)
23. Kim, Y.: A new branch and bound algorithm for minimizing mean tardiness in two-machine flowshops. Computers & Operations Research 20(4), pp. 391–401 (1993)
24. Parthasarathy, S., Rajendran, C.: A simulated annealing heuristic for scheduling to minimize mean weighted tardiness in a flowshop with sequence-dependent setup times of jobs – A case study. Production Planning and Control 8, pp. 475–483 (1997)
25. Framinan, J. M., Leisten, R.: Total tardiness minimization in permutation flowshops: a simple approach based on a variable greedy algorithm. International Journal of Production Research 46(22), pp. 6479–6498 (2008)

26. Vallada, E., Ruiz, R., Minella, G.: Minimizing total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. Computers & Operations Research 35(4), p. 1350 (2008)
27. Liu, W., Tony, T., Xia, Y.: Two-sample test of high dimensional means under dependence. Journal of the Royal Statistical Society 76(2), pp. 349–372 (2014)
28. Abdul-Razaq, T. S., Potts, C. N., Van Wassenhove, L. N.: A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete Applied Mathematics 26(2–3), pp. 235–253 (1990)
29. Resende, M. G., Ribeiro, C. C.: A GRASP with path relinking for private virtual circuit routing. Networks 41(2), pp. 104–114 (2003)
30. Mahinthakumar, K., Sayeed, M.: Hybrid Genetic Algorithm - Local Search Methods for Solving Groundwater Source Identification Inverse Problem. Journal of Water Resources Planning and Management 131(1) (2005)
31. OR-Library, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/wtinfo.html (2018)