Revista Técnica de la Facultad de Ingeniería, Universidad del Zulia
ISSN: 0254-0770
revistatecnica@fing.luz.edu.ve
Universidad del Zulia
República Bolivariana de Venezuela

Juan Paredes-Quevedo; Luis Alpala; Luis Soto-Chávez; Alberto León-Batallas

Evaluación del Algoritmo Genético y GRASP para Minimizar el Makespan en la Programación de un Taller de Flujo en Diferentes Instancias de Número de Trabajos e Iteraciones

Revista Técnica de la Facultad de Ingeniería,
Universidad del Zulia, vol. 45, núm. 1, 2022, pp. 48-57
Universidad del Zulia
Maracaibo, República Bolivariana de Venezuela

DOI: https://doi.org/10.22209/rt.v45n1a05

Disponible en: https://www.redalyc.org/articulo.oa?id=605780378005

Evaluación del Algoritmo Genético y GRASP para Minimizar el *Makespan* en la Programación de un Taller de Flujo en Diferentes Instancias de Número de Trabajos e Iteraciones

Juan Paredes-Quevedo[1]*, Luis Alpala[2,3], Luis Soto-Chávez[4], Alberto León-Batallas[1]

[1]Facultad Ciencias e Ingeniería, Universidad Estatal de Milagro (UNEMI), Km 1.5 vía Km 26, Ciudadela Universitaria Milagro 09150, Ecuador.
[2]Grupo de Investigación Eslinga. Universidad Cooperativa de Colombia-Pasto, Colombia.
[3]Grupo de Investigación en Informática Gráfica. Universidad de Granada, España.
[4]Facultad de Ingeniería Industrial. Universidad de Guayaquil, Ecuador.
*Autor de correspondencia: jparedesq@unemi.edu.ec

https://doi.org/10.22209/rt.v45n1a05

Recepción: 18 de junio 2020 | Aceptación: 15 de octubre de 2021 | Publicación: 27 de diciembre de 2021

Resumen

En este artículo se plantean algoritmos meta-heurísticos para el problema de programación de producción de un taller de flujo, considerado como un problema no polinómico completo debido a su complejidad. El estudio de la problemática es relevante dada su utilidad en la práctica, por ejemplo, en fábricas con líneas de ensamblaje o en la planificación de cadenas de suministro colaborativas. De esta manera, el objetivo del presente estudio consistió en evaluar dos algoritmos meta-heurísticos, GRASP y algoritmo genético. Se planteó un diseño experimental para determinar cuáles factores (método, número de trabajos y número de iteraciones) tienen un efecto estadísticamente significativo en el tiempo de finalización de programación de producción. Según los resultados se pudo observar que existe incidencia tanto de primer como de segundo y tercer orden de los factores, permitiendo caracterizar el desempeño de los dos algoritmos en entornos generados por la interacción de los tres factores analizados.

Palabras clave: anova; algoritmo genético; GRASP; makespan; taller de flujo.

Evaluation of Genetic Algorithm and GRASP to Minimize Makespan in Flow Shop Scheduling at Different Instances of Number of Jobs and Iterations

Abstract

In this article, meta-heuristic algorithms are proposed for the production scheduling problem of a flow workshop, considered as a complete non-polynomial problem due to its complexity. The study of the problem is relevant given its usefulness in practice, for example, in factories with assembly lines or in collaborative supply chains planning. Thus, the objective of the present study consisted of evaluating two meta-heuristic algorithms, GRASP and a genetic algorithm. An experimental design was proposed to determine which factors (method, number of jobs and number of iterations) have a statistically significant effect on production scheduling completion time. According to the results, it could be observed that there is incidence of the first, second and third order of the factors, allowing to characterize the performance of the two algorithms in environments generated by the interaction of the three factors analyzed.

Key words: anova; genetic algorithm; GRASP; makespan; flow shop.

Avaliação do Algoritmo Genético e GRASP para Minimizar o Makespan na Programação de uma Oficina de Fluxo em Diferentes Instâncias de Número de trabalhos e Iterações

Resumo

Neste artigo, algoritmos meta-heurísticos são propostos para o problema de programação de produção de uma oficina de fluxo, considerado um problema não polinomial completo devido à sua complexidade. O estudo do problema é relevante dada a sua utilidade na prática, por exemplo, em fábricas com linhas de montagem ou no planejamento de cadeias produtivas colaborativas. Assim, o objetivo deste estudo consistiu em avaliar dois algoritmos meta-heurísticos, GRASP e um algoritmo genético. Um projeto experimental foi proposto para determinar quais fatores (método, numero de trabalhos e iterações) têm um efeito estatisticamente significativo no tempo de conclusão do cronograma de produção. De acordo com os resultados, observou-se que há incidência tanto da primeira, segunda e terceira ordem dos fatores, permitindo caracterizar o desempenho dos dois algoritmos, em ambientes gerados pela interação dos três fatores analisados.

Palavras-chave: anova; algoritmo genético; GRASP; makepan; oficina de fluxo.

Introducción

El *makespan* o *Cmáx* es el criterio de optimización más estudiado en la literatura reciente (Lagos *et al*., 2019; Castillo *et al*., 2018), a través de la minimización del tiempo máximo de terminación de la programación (Vallada y Ruiz, 2011). Se considera que minimizar el *makespan* equivale a maximizar la utilización de las máquinas, en otras palabras, disminuir los tiempos de alistamiento y de ocio de las máquinas (Hekmatfar *et al*., 2011).

Los problemas de programación de producción en su mayoría son no polinómicos (NP)-completos y requieren para su solución procedimientos complejos y costosos en el tiempo (Tavares Neto y Godinho Filho, 2013). Por ello, existen muchas técnicas como las heurísticas o metaheurísticas que surgen como alternativas interesantes, para encontrar soluciones de muy buena calidad, en tiempos de cómputo bastante razonables (Hussain *et al*., 2018).

Para esta investigación se evaluaron dos metaheurísticas como son GRASP (*Greedy Randomized Adaptive Search Procedures*) y algoritmo genético; estos dos algoritmos tienen diferentes procedimientos para llegar a la solución más óptima. La metaheurística GRASP es un método que construye una solución miope aleatorizada; más una búsqueda local que se ejecuta utilizando la solución construida como punto inicial de partida (Resende y González, 2003).

Taller de flujo – *flow shop*

En el tipo de configuración productiva del caso general *flow shop*, el problema de programación consiste en secuenciar una cantidad de *n* trabajos, que siguen la misma ruta de procesamiento, en una serie de *m* máquinas ordenadas linealmente (Akhshabi *et al*., 2012; Shabtay, 2012). La programación de producción del *flow shop* es un problema combinatorio bastante complejo, ya que la cantidad de posibles alternativas de secuenciación requeriría varios años si se quieren enumerar todas las posibilidades de un problema grande (Hentous y Merabti, 2010).

Metaheurísticas aplicadas a la dirección de producción y operaciones

Los problemas de programación de producción son en su mayoría NP-completos y requieren para su solución, de procedimientos complejos y costosos en el tiempo (Tavares Neto y Godinho Filho, 2013). En casos como este, las técnicas de optimización combinatorial surgen como alternativa interesante, ya tienen la capacidad de encontrar soluciones de muy buena calidad, en tiempos de cómputo bastante razonables (Villalba *et al*., 2005). Los métodos de resolución existentes en la literatura pueden agruparse en dos grandes familias, que

son: los métodos exactos (como el modelo de programación lineal entera) y los métodos de aproximación (como las heurísticas y metaheurísticas) (Yalaoui *et al*., 2011; Lagos *et al*., 2019).

La complejidad computacional y del modelamiento que implica el problema de programación del *flow shop*, conlleva a que el problema de optimización combinatorial resulte muy complicado y difícil (Zhang, 2010).

Actualmente, las metaheurísticas más aplicadas a los problemas de programación de configuraciones tipo *flow shop* son el algoritmo de búsqueda tabú, la optimización por colonia de hormigas, los algoritmos genéticos, el recocido simulado, entre otros. Cada uno de los cuales tiene sus propias ventajas y desventajas (Qiao y Sun, 2011).

Algoritmos genéticos

Los algoritmos genéticos, desarrollados en la década de los 70 por Holland (Gómez, 2007; Mahdavi *et al*., 2008), son técnicas de búsqueda heurística que toman la analogía de los conceptos de la selección natural, empleando una población de soluciones candidatas y combinándolas en formas específicas con el fin de obtener mejores soluciones (Feng *et al*., 2009). Estos algoritmos se han convertido en una metodología muy popular para solucionar una gran variedad de problemas complejos (Bertel y Billaut, 2004). El algoritmo genético, siendo una técnica de búsqueda estocástica, se ha logrado aplicar a varias áreas, incluyéndose los problemas de programación de máquinas (Hsu *et al*., 2009).

En el algoritmo genético una solución factible se conoce con el término de cromosoma, el cual es representado con una cadena de valores enteros, cada uno de los cuales se denomina gen (Tang *et al*., 2010). La calidad de un cromosoma se denomina *fitness*, que establece la concordancia de la alternativa de solución de acuerdo con la función objetivo definida (Chiou *et al*., 2012). Las combinaciones de los genes van evolucionando a través de las operaciones genéticas (reproducción, cruce y mutación) para crear la descendencia, de manera que los cromosomas se van aproximando a la solución óptima generación tras generación (Engin *et al*., 2011). Sin embargo, es necesaria una codificación adecuada para cada problema y tener una función de ajuste que represente claramente la medida de calidad para cada solución alternativa (Hekmatfar *et al*., 2011).

Metaheurísticas GRASP

Es una técnica combinatorial que permite encontrar soluciones subóptimas de buena calidad a problemas de optimización lineales o no lineales. GRASP es una metodología que construye una solución inicial mediante una función de adaptación codiciosa o *greedy* aleatoriamente (Ángel-Bello *et al*., 2011). Posteriormente, una búsqueda local se ejecuta utilizando una solución construida como punto inicial de partida. Una solución se construye mediante la incorporación de un nuevo elemento al tiempo hasta que una nueva solución se ha completado (Resende y Ribeiro, 2016).

Una de las principales características de este método, es el uso de las listas restringidas de candidatos (LRC). Una LRC contiene los elementos candidatos con mejor valor de función *greedy*, que ayuda a enfocar más el proceso de búsqueda. Los ciclos iterativos se llevarán a cabo hasta que se cumpla cierto número de búsquedas en la vecindad, o se haya explorado toda la región de vecindad o cuando no se supere la solución actual en un determinado número de exploraciones (Resende y Ribeiro, 2016).

Materiales y Métodos

El diseño experimental propuesto nos condujo a conocer cuál método es mejor y bajo qué condiciones. El primer paso del diseño de experimentos es tener claro el método de análisis que se empleará. Para este caso, se quiere determinar la influencia de las variables explicativas con respecto al "incremento porcentual sobre el óptimo del valor medio examinado " (IPSOVEP), tras comparar dos metaheurísticas (GRASP y algoritmo genético), en diferentes escenarios en torno al número de trabajos e iteraciones. El método de análisis (anova o análisis de varianza) mediante factores múltiples.

Identificación de la variable explicada y de las variables explicativas

Para el estudio se definió una variable explicada (dependiente) cuyo valor es función de varias variables explicativas (independientes), a saber:

- Variable explicada: IPSOVEP

- Variables explicativas: método (Mt), número trabajos (nT), iteraciones (It).

Como medida comparativa (variable explicada) se utilizó la media del aumento de porcentaje sobre el nivel óptimo, o el nivel más bajo conocido del resultado medio (IPSOVEP medio) de un problema o instancia determinada. Esta medida se puede expresar, como (Gómez-Gasquet *et al*., 2012):

$$\text{IPSOVEP} = \left(\sum_{i=1}^{n} \frac{\text{Sol}_i - \text{MejorSol}_i}{\text{MejorSol}_i}\right)/n \quad (1)$$

La variable Sol_i es el valor del *makespan* obtenido con una instancia dada del algoritmo en evaluación. La variable MejorSol_i representa el valor del mejor *makespan* conocido para esta instancia. Por lo tanto, los valores positivos para IPSOVEP implican que el algoritmo tenga un margen de tiempo peor que el punto de referencia utilizado, y los valores negativos para IPSOVEP implican que el modelo ha sido mejorado. Finalmente, la variable n es el número de ejemplares representativos que se consideren para calcular el valor del IPSOVEP medio.

Estrategia experimental o plan de experimentación

El planteamiento se realizó con un conjunto de 10 ejemplares, para cada instancia de datos que se empleó para la ejecución del algoritmo. Posteriormente, se calculó el valor medio IPSOVEP del conjunto de ejemplares asociado a cada instancia. Este procedimiento se repitió hasta tener una muestra de diez datos.

Cada instancia era una combinación de parámetros que define un escenario y estaba vinculado a los valores que toman las variables explicativas. Variables explicativas (factores):

- Método (mT) = algoritmo genético, GRASP.
- Número de trabajos (nT) = 20, 50.
- Iteraciones (iT) = 1000, 10000.

La combinación de parámetros para el escenario propuesto fue de: 2 x 2 x 2 = 8 combinaciones.

Algoritmo propuesto

La metaheurística GRASP es de tipo multi-arranque y está provista de dos fases en cada iteración (Resende y González, 2003): una primera fase un procedimiento *greedy* que sirve para construir una solución aceptable sin que sea preciso alcanzar el óptimo global, y una segunda fase para obtener un óptimo local dentro de un vecindario y teniendo como punto de partida la solución que resulta al aplicar el procedimiento *greedy* de la fase uno. En la Figura 1 se puede observar un esquema general del algoritmo (Bautista *et al*., 2013).

1. Inicialización
2. Mientras tiempo CPU ≤ CPU_{max}
   2.1. Solución ← Fase_constructiva (*Semilla*)
   2.2. Solución ← Mejora_local (*Solución*)
   2.3. Acutalizar_solución (*Solución*, *Mejor_solución*)
3. Salida: *Mejor_solución*

Figura 1. Esquema general de la meta-heurística GRASP (Bautista *et al*., 2013). CPU: *central processing unit.*

Fase constructiva: para garantizar dicha diversidad se empleó el azar, de manera que el siguiente elemento a añadir a una solución parcial se elige aleatoriamente entre los elementos de una lista restringida de candidatos (LRC); dicha lista contenía los candidatos que presentan los mejores valores con relación a una función objetivo diseñado para la selección.

Para el presente trabajo se ha considerado utilizar el mecanismo de construcción con LRC basada en el valor (Resende y Ribeiro, 2016) (Figura 2). Esta LRC contenía todos los elementos candidatos cuyo valor de función objetivo C(e) cumplía con el siguiente criterio: $C(e) <= C_* + \alpha (C^* - C_*)$. Se puede observar que si $\alpha = 0$, entonces este esquema de selección es un algoritmo miope, mientras que si $\alpha = 1$, entonces es totalmente aleatorio.

```
procedure Construcción-V
Require: α, E, c(·);
    x ← ∅;
    C ← E;
    Calcular costo miope c(e), ∀ e ∈ C;
    While C ≠ ∅ do
        c* ← mín {c(e) | e ∈ C};
        Actualizar el conjunto de candidatos C;
        Calcular el costo miope c(e), ∀ e ∈ C;
    end while
        c* ← máx {c(e) | e ∈ C};
        RCL ← {e ∈ C |c(e) ≤ c* + α(c* − c*)};
        Seleccionar un elemento s de LRC al azar;
        x ← x ∪ {s};
    return x;
```

Figura 2. Seudo-código de construcción GRASP: LCR basada en valor (Resende y González, 2003). c(e): función miope (*Cmáx*, por ejemplo) dado un elemento candidato, C: conjunto de elementos que pueden seleccionarse como parte de una solución parcial, α: número aleatorio que oscila entre 0 y 1.

Fase de búsqueda local: se proponen dos procedimientos para ejecutar la fase de búsqueda local. El primer procedimiento de mejora local es un *adJointSwap* que se aplica a la solución obtenida en la fase constructiva. El segundo procedimiento (*Swap* + *adJointSwap*) consistió en un bucle que aplica consecutivamente una mutación *Swap* a la solución obtenida en el primer procedimiento hasta alcanzar un óptimo local, y seguidamente se procede a realizar una búsqueda local con una mutación *adJointSwap* (primer procedimiento).

Tanto para el primer como para el segundo procedimiento, se consideró definir los parámetros respectivos (Tabla 1).

Tabla 1. Número máximo de iteraciones para la fase de búsqueda local.

Parámetro	Valor
Condición de parada *adJointSwap*	1000
Condición de parada *Swap*	1000

Desarrollo de los experimentos

Para la ejecución de los experimentos se utilizó el programa Java versión 2017, donde se obtuvieron los ficheros de resultados. Para el tratamiento de los resultados se empleó el programa estadístico Statgraphics versión 2017, realizando el análisis anova multifactorial.

Resultados y Discusión

En el experimento computacional se usaron las instancias del problema descritas anteriormente. Los resultados del IPSOVEP medio de cada una de las 8 instancias y de las 10 muestras, se visualizan en la Tabla 2. Para la obtención del valor IPSOVEP se tomaron los valores óptimos proporcionados por Taillard (1993); los cuales también están presentes en estudios realizados por Ocampo *et al*. (2006) y Gómez-Gasquet *et al*. (2012).

En la Tabla 2 se compara el valor de IPSOVEP obtenido tanto para el algoritmo genético como para GRASP, considerando las instancias de problemas con 20 y 50 trabajos, así como 1000 y 10000 iteraciones para buscar la mejor solución.

De estos resultados se puede hacer las siguientes observaciones: en general, ambos algoritmos presentan un desempeño similar (la media de las muestras: 7,70 algoritmo genético y 7,53 GRASP), cuando el número de trabajos es menor (20) y el número de iteraciones es igual a 1000.

Tabla 2. Comparativa de IPSOVEP obtenido para cada algoritmo.

N° de muestra	Número de trabajos (nT)	Número de iteraciones (iT)	IPSOVEP	
			AG	GRASP
0	20	1000	8,14623	7,85123
1	20	1000	7,58204	7,60187
2	20	1000	7,32192	5,77536
3	20	1000	7,11603	6,79938
4	20	1000	7,37709	7,46124
5	20	1000	7,74586	7,24190
6	20	1000	7,68815	7,82348
7	20	1000	8,58171	8,48440
8	20	1000	7,89753	8,60354
9	20	1000	7,61444	7,69766
0	50	1000	4,20164	3,39033
1	50	1000	4,61916	2,82515
2	50	1000	4,03438	3,43772
3	50	1000	5,08509	3,08596
4	50	1000	4,73249	3,09694
5	50	1000	4,64585	3,35456
6	50	1000	4,97437	2,96714
7	50	1000	4,92010	3,11762
8	50	1000	4,87561	2,79198
9	50	1000	4,64119	2,87680
0	20	10000	6,03736	7,58858
1	20	10000	6,15794	9,16808
2	20	10000	6,10841	10,19786
3	20	10000	6,71705	7,33776
4	20	10000	6,17980	7,83965
5	20	10000	6,46187	8,32607
6	20	10000	6,45557	7,41847
7	20	10000	6,27003	8,33910
8	20	10000	6,42008	6,96850
9	20	10000	6,99880	8,06642
0	50	10000	4,32398	3,08958
1	50	10000	4,01301	3,02947
2	50	10000	4,15659	2,58577
3	50	10000	3,74023	3,49007
4	50	10000	4,06964	3,14456
5	50	10000	4,30636	3,52381
6	50	10000	3,74479	3,53161
7	50	10000	3,96751	2,99282
8	50	10000	4,13869	2,30381
9	50	10000	3,98973	3,25579

IPSOVEP: incremento porcentual sobre el óptimo del valor medio examinado, AG: algoritmo genético, GRASP: *greedy randomized adaptive search procedures.*

En la instancia de 50 trabajos y 1000 iteraciones se visualiza que el algoritmo GRASP tuvo un valor medio de 3,09 y el algoritmo genético 4,67, es decir, el primero tuvo mejor desempeño en esta instancia.

El algoritmo genético presentó una pequeña ventaja cuando se experimentaba con 20 trabajos y 10000 iteraciones, dado que obtuvieron valores medios de IPSOVEP de 6,38 y de algoritmo GRASP de 8,12.

Para un análisis resumido se generó la Figura 3 (interacciones entre mT – nT) donde se puede observar que el algoritmo GRASP cuando el número de trabajos es igual a 20 registra un IPSOVEP mayor con respecto al valor registrado por el algoritmo genético. En el mismo gráfico se puede ver que al tratarse de un número de trabajos mayor (50) el algoritmo GRASP registra un valor de IPSOVEP menor que el algoritmo genético.

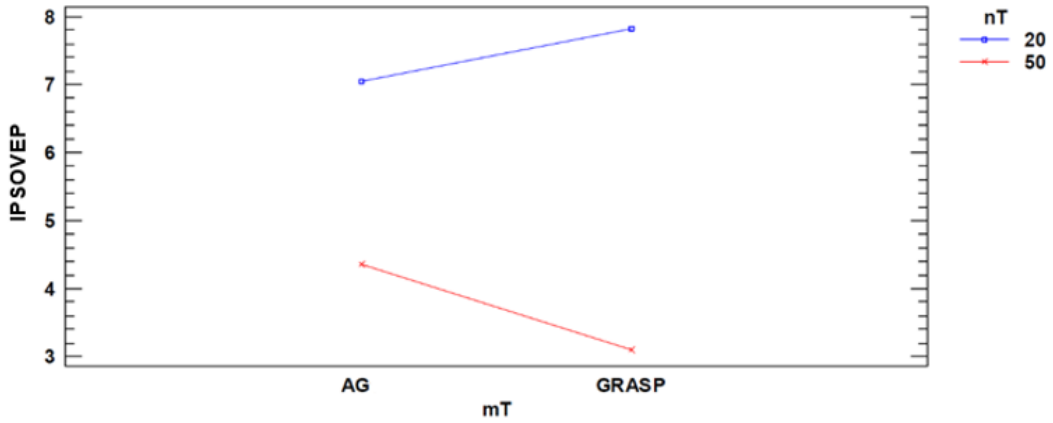

Figura 3. IPSOVEP medio para 20 y 50 trabajos. nT: número de trabajos, mT: algoritmos analizados, IPSOVEP: incremento porcentual sobre el óptimo del valor medio examinado, GRASP: *greedy randomized adaptive search procedures*, AG: algoritmo genético.

Por otra parte, en la Figura 4 (interacciones entre mT- iT), se puede observar que el algoritmo GRASP cuando el número de iteraciones es igual a 1000 registra un IPSOVEP menor con respecto al valor registrado por el algoritmo genético. En el mismo gráfico se puede ver que al tratarse de un número de iteraciones mayor (10000) el algoritmo GRASP registra un valor de IPSOVEP mayor que el algoritmo genético.

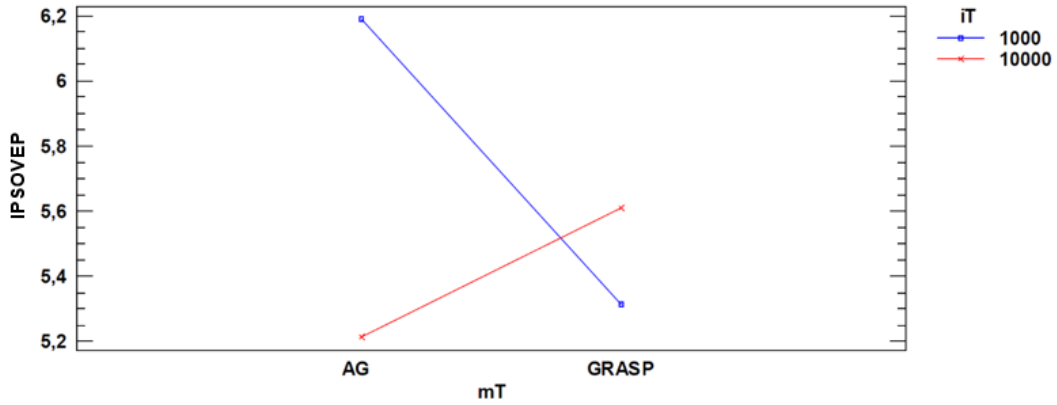

Figura 4. IPSOVEP medio para 1000 y 10000 iteraciones. iT: número de iteraciones, mT: algoritmos analizados, IPSOVEP: incremento porcentual sobre el óptimo del valor medio examinado, GRASP: *greedy randomized adaptive search procedures*, AG: algoritmo genético.

En la Tabla 3 se muestran los resultados del anova, resultantes del análisis multifactorial para IPSOVEP y las combinaciones significativas con un α=0,05. Los valores-P fueron menores que 0,05; estos factores de manera individual tienen un efecto estadísticamente significativo sobre IPSOVEP con un 95,0 % de nivel de confianza. Adicionalmente, se valida el análisis de los 3 factores, dado que la interacción conjunta tiene efecto significativo sobre el valor del IPSOVEP, lo cual implica un insumo para la toma de decisión al momento de elegir un algoritmo para determinada instancia.

Tabla 3. Resultados del análisis de varianza para IPSOVEP (suma de cuadrados tipo III).

Fuente	Suma de cuadrados	gl	Cuadrado medio	Razón-F	Valor-P
Efectos principales					
A:mT	1,14631	1	1,14631	4,12	0,0461
B:nT	275,269	1	275,269	989,07	0,0000
C:iT	2,32219	1	2,32219	8,34	0,0051
Interacciones					
AB	21,0141	1	21,0141	75,51	0,0000
AC	8,10071	1	8,10071	29,11	0,0000
BC	0,0145086	1	0,0145086	0,05	0,8200
ABC	2,07757	1	2,07757	7,46	0,0079
Residuos	20,0383	72	0,27831		
Total (corregido)	329,983	79			

nT: número de trabajos, mT: algoritmos analizados, nI: número de iteraciones, gl: grados de libertad.

El algoritmo metaheurístico GRASP propuesto incluye en la fase de búsqueda local, dos procedimientos para mejorar la solución obtenida en la fase constructiva, validada a través de los estudios revisados, que analizan el problema de programación de un taller de flujo (Zobolas *et al*., 2009; Salazar Hornig y Figueroa Morales, 2012; César *et al*., 2014) y que utilizan los datos proporcionados por Taillard (1993).

Del mismo modo, se pueden encontrar algoritmos GRASP con la misma función objetivo que el algoritmo propuesto (Prabhaharan et al., 2006; Bautista et al., 2013), y con multiobjetivo (Choong y Alias, 2011; Zabihzadeh y Rezaeian, 2016), que nos permiten validar el rendimiento del algoritmo GRASP propuesto en este trabajo. A su vez, los resultados revisados muestran valores porcentuales con respecto al óptimo que no presentan diferencias considerables. Motivo por el cual, el algoritmo propuesto de baja complejidad en su construcción representa una alternativa viable para la solución de la problemática de programación de un taller de flujo.

Conclusiones

Al evaluar el desempeño de los algoritmos planteados en el problema de programación de producción de un *flow shop* no se puede afirmar de manera absoluta que uno de los algoritmos proporcione los mejores resultados que el otro, dado que en las diferentes instancias determinadas por la combinación de los factores (variables explicativas) los valores de IPSOVEP de dichos algoritmos varían.

El procedimiento anova multifactorial permitió construir un modelo estadístico para conocer si existe un impacto significativo de los factores (mT, nT, iT) en una variable dependiente IPSOVEP. Según los resultados se pudo observar que existe incidencia tanto de primer, segundo y tercer orden de los factores sobre la variable dependiente. Esto permite caracterizar de manera más precisa el comportamiento y desempeño de estos algoritmos en entornos generados por la interacción de los tres factores analizados en este artículo. En la práctica esta información puede permitir mejorar el proceso de la toma de decisiones en la planta.

La investigación futura podría abordar este enfoque para problemas más complejos de *flow shop*, que involucren tiempos de alistamientos dependientes de la secuencia. También se pueden probar con diferentes funciones objetivo.

Referencias Bibliográficas

Akhshabi, M., Haddadnia, J., Akhshabi, M. (2012). Solving flow shop scheduling problem using a parallel genetic algorithm. *Procedia Technology*, 1, 351-355.

Ángel-Bello, F., Álvarez, A., Pacheco, J., Martínez, I. (2011). A heuristic approach for a scheduling problem with periodic maintenance and sequence-dependent setup times. *Computers and Mathematics with Applications*, 61, 797-808.

Bautista, J., Cano, A., Alfaro, R., Batalla, C. (2013). *Algoritmos GRASP para solucionar el problema blocking flow shop.* Multiconferencia CAEPIA'13. IX congreso español de metaheurísticas, algoritmos evolutivos y bioinspirados - MAEB 2013. Barcelona: Universitat Politécnica de Catalunya, 443-452.

Bertel, S., Billaut, J.C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *Journal of the Operational Research Society*, 159, 651-662.

Castillo, T. A., Díaz, C. E., Gómez, J. D., Orduz, E. A., Niño, M. L. (2018). Optimización del makespan en el problema de *Job Shop Flexible* con restricciones de transporte usando Algoritmos Genéticos. *Entre Ciencia e Ingeniería*, 12, 105-115.

César, Y., Reyna, F., Jiménez, Y.M., Enrique, Á., León, F., Alberto, L., 2014. Influencia de los parámetros principales de un Algoritmo Genético para el Flow Shop Scheduling. *Revista Cubana de Ciencias Informáticas*, 8, 99–111.

Chiou, C. W., Chen, W. M., Liu, C. M., Wu, M. C. (2012). A genetic algorithm for scheduling dual flow shops. *Expert Systems with Applications*, 39, 1306-1314.

Choong, F., Alias, M.Y., 2011. Expert Systems with Applications Metaheuristic methods in hybrid flow shop scheduling problem. *Expert Systems With Applications*, 38, 10787–10793.

Engin, O., Ceran, G., Yilmaz, M. K. (2011). An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. *Applied Soft Computing Journal*, 11, 3056-3065.

Feng, H., Lu, S., Li, X. (2009). *Genetic algorithm for hybrid flow-shop scheduling with parrel batch processors.* 2009 WASE international conference on information engineering, ICIE 2009. Jinan: IEEE Computer Society, 2, 9-13.

Gómez-Gasquet, P., Andrés, C., Lario, F. C. (2012). An agent-based genetic algorithm for hybrid flowshops with sequence dependent setup times to minimise makespan. *Expert Systems with Applications*, 39, 8095-8107.
Gómez, P. (2007). *Un nuevo algoritmo genético basado en un sistema multiagente para la programación de la producción en un taller de flujo híbrido.* XI congreso de ingeniería de organización. Madrid: Asociación para el Desarrollo de la Ingeniería de Organización,1675-1685.

Hekmatfar, M., Fatemi Ghomi, S. M. T., Karimi, B. (2011). Two stage reentrant hybrid flow shop with setup times and the criterion of minimizing makespan. *Applied Soft Computing Journal*, 11, 4530-4539.

Hentous, H., Merabti, B. (2010). *A branch and bound heuristic for the flow shop problem*. Proceedings - 4th international conference on sensor technologies and applications, SENSORCOMM 2010. Venice: IEEE Computer Society, 352-356.

Hsu, H. M., Hsiung, Y., Chen, Y. Z., Wu, M. C. (2009). A GA methodology for the scheduling of yarn-dyed textile production. *Expert Systems with Applications*, 36, 12095-12103.

Hussain, K., Najib, M., Salleh, M., Cheng, S., Shi, Y. (2018). Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52, 2191-2233.

Lagos, D., Mancilla, R., Leal, P. (2019). Evaluación de la solución obtenida con una metaheurística de secuenciamiento mediante la incorporación de elementos de variabilidad. *Revista Técnica de la Facultad de Ingeniería, Univiversidad del Zulia*, 1, 209-212.

Mahdavi, I., Mojarad, M.S., Javadi, B., Tajdin, A. (2008). *A genetic approach for solving a hybrid flow shop scheduling problem*. 2008 IEEE international conference on industrial engineering and engineering management, IEEM 2008. Singapore: IEEE, 1214-1218.

Ocampo, T., Mirledy, E., Grisales, R., Steven, Y. O. V., Echeverri, G. (2006). Algoritmo genético modificado aplicado al problema de secuenciamiento de tareas en sistemas de producción lineal - *Flow shop*. *Scientia Et Technica*, 7, 285-290.

Prabhaharan, G., Khan, S.H., Rakesh, L., 2006. Implementation of grasp in flow shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 30, 1126–1131.

Qiao, P., Sun, C. (2011). *Research on hybrid flow-shop scheduling problem based on improved immune particle swarm optimization.* 2nd international conference on artificial intelligence, management science and electronic commerce, AIMSEC. Deng Feng: IEEE, 4240-4243.

Resende, M. G. C., González, J. L. (2003). GRASP: Procedimientos de búsquedas miopes aleatorizados y adaptativos. Inteligencia artificial. *Revista Iberoamericana de Inteligencia Artificial*, 7, 61-76.

Resende, M. G. C., Ribeiro, C. C. (2016). *Optimization by GRASP*. 1st ed. New York: Springer Science+Business Media.

Salazar Hornig, E., Figueroa Morales, B., 2012. Minimización de la tardanza para el flowshop flexible con setup utilizando heurísticas constructivas y un algoritmo genético. *Ingeniare. Revista chilena de ingeniería*, 20, 89–98.
Shabtay, D. (2012). The just-in-time scheduling problem in a flow-shop scheduling system. *European Journal of Operational Research*, 216, 521-532.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2), 278-285.

Tang, J., Zhang, G., Lin, B., Zhang, B. (2010). *Hybrid genetic algorithm for flow shop scheduling problem*. International conference on intelligent computation technology and automation, ICICTA 2010. Changsha: IEEE, 2, 449-452.

Tavares Neto, R.F., Godinho Filho, M. (2013). Literature review regarding ant colony optimization applied to scheduling problems: Guidelines for implementation and directions for future research. *Engineering Applications of Artificial Intelligence*, 26, 150-161.

Vallada, E., Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research*, 211, 612-622.

Villalba, G., Castro, F. De, Saldarriaga, J. G. (2005). Algoritmos de optimización combinatoria (AOC) aplicados al diseño de redes de distribución de agua potable. *Revista de Ingeniería Universidad de los Andes*, 22, 118-125.

Yalaoui, N., Mahdi, H., Amodeo, L., Yalaoui, F. (2011). A particle swarm optimization under fuzzy logic controller to solve a scheduling problem. *International Conference on Communications, Computing and Control Applications (CCCA)*, Hammamet: IEEE, 1–6.

Zabihzadeh, S.S., Rezaeian, J., 2016. Two meta-heuristic algorithms for flexible flow shop scheduling problem with robotic transportation and release time. *Applied Soft Computing Journal*, 40, 319–330.

Zhang, S. (2010). *Large-scale flow shop scheduling based on genetic algorithm*.. 2nd international conference on education technology and computer, ICETC 2010. Shanghai: IEEE, 1, 308-310.

Zobolas, G.I., Tarantilis, C.D., Ioannou, G., 2009. Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. *Computers & Operations Research*, 36, 1249–1267.