J. C. Seck-Tuoh-Mora,* G. E. Anaya-Fuentes,* N. Hernández-Romero,*
J. Medina-Marín,* I. Barragán-Vite,* M. A. López-Cabrera*

Optimización de trabajadores y estaciones de trabajo en líneas de ensamble multi-tripuladas mediante algoritmos genéticos◊

Optimization of workers and workstations in multi-manned assembly lines using genetic algorithms

Abstract | Assembly lines are a production mechanism that has historically presented economic and quality benefits in organizations. However, different problems arise during their execution, among them, the issue of balancing assembly lines with multi-manned stations. This situation occurs more frequently in industrial organizations that manufacture products of medium and large dimensions, compared to a single worker per workstation model. Despite this, we find a greater tendency to study the second case in the literature. In contrast, few studies refer to the first, in addition, exhaustive search methods such as linear programming have encountered barriers due to computational complexity, so research on this problem has focused on using heuristics looking for more efficient algorithms in order to solve it. Therefore, the present work proposes a genetic algorithm that, to our knowledge, has not been used in the search to minimize the number of workers and the number of workstations for the balancing model of assembly lines with multi-manned stations. In addition, a new cost function is proposed that weights the number of workstations and workers, punishing solutions with high idle times to avoid their selection. The results of the proposed algorithm are evaluated by comparing test instances presented in the literature. The algorithm is available at <https://github.com/juanseck/GAMmALB>.

Keywords | bi-objective optimization | assembly line balancing | genetic algorithms | multi-manned workstations | costs.

Recibido: 26 de agosto, 2022. Aceptado: 13 de diciembre, 2022.
◊ Este estudio ha sido realizado gracias al apoyo del Conacyt, con números de proyecto CB- 2017-2018-A1-S-43008 y F003/320109, y con el respaldo de la Universidad Autónoma del Estado de Hidalgo.
* Universidad Autónoma del Estado de Hidalgo, Instituto de Ciencias Básicas e Ingeniería, Área Académica de Ingeniería.
Correo electrónico: gustavoerick_anay@hotmail.com

Seck-Tuoh-Mora, J. C., G. E. Anaya-Fuentes, N. Hernández-Romero, J. Medina-Marín, I. Barragán-Vite, M. A. López-Cabrera. «Optimización de trabajadores y estaciones de trabajo en líneas de ensamble multi-tripuladas mediante algoritmos genéticos.» *INTER DISCIPLINA* 12, nº 33 (mayo–agosto 2024): 59-83.
DOI: https://doi.org/10.22201/ceiich.24485705e.2024.33.88239

Resumen | Las líneas de ensamble son un mecanismo de producción que históricamente ha presentado beneficios económicos y de calidad en las organizaciones; sin embargo, diferentes problemas se presentan durante su ejecución. Entre estos destaca el problema del modelo de balanceo de líneas de ensamble con estaciones multi-tripuladas, el cual suele presentarse con mayor frecuencia en las organizaciones industriales que fabrican productos de dimensiones medianas y grandes, en comparación con el modelo de un solo tripulante por estación de trabajo; a pesar de ello, en la literatura encontramos una mayor tendencia a estudiar el segundo caso. Pocos estudios hacen referencia al primero. Adicionalmente, los métodos de búsqueda exhaustiva como los de programación lineal han encontrado barreras debido a la complejidad computacional, por lo cual las investigaciones sobre este problema se han centrado en utilizar heurísticos para su solución, en la búsqueda de algoritmos que sean cada vez más eficientes. Por lo anterior, el presente trabajo propone un algoritmo genético que en nuestro conocimiento no se ha utilizado en la búsqueda por minimizar el número de trabajadores y el número de estaciones de trabajo para el modelo de balanceo de líneas de ensamble con estaciones multi-tripuladas. Además, se propone una nueva función de costos la cual pondera el número de estaciones de trabajo y trabajadores, castigando soluciones con tiempos muertos altos para evitar su selección. Los resultados del algoritmo propuesto son evaluados mediante la comparación de instancias de prueba presentadas en la literatura. El algoritmo está disponible en <https://github.com/juanseck/GAMmALB>.
Palabras clave | optimización bi-objetivo | balanceo de líneas de ensamble | algoritmos genéticos | estaciones de trabajo multi-tripuladas | costos.

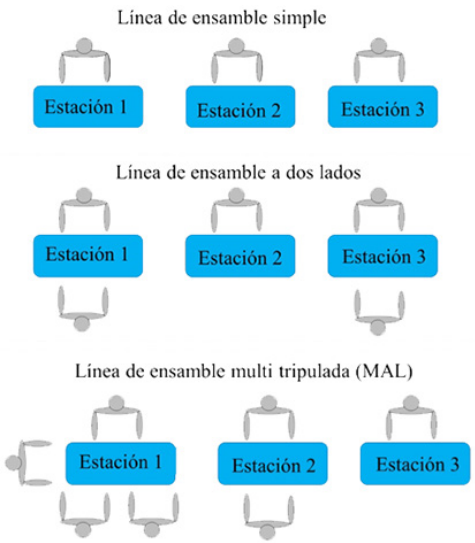
Introducción

LA COMPETITIVIDAD GENERADA por la globalización obliga a las organizaciones con fines de lucro a buscar mecanismos y estrategias que les mantengan en la participación del mercado. Indicadores como la producción y los costos se vinculan y pueden asociarse con un indicador único, la productividad, entendida como el cociente de lo producido y los requerimientos. En este sentido, el objetivo primordial para los directivos de las organizaciones es la minimización de los costos de producción. Para minimizar costos, existen diferentes estrategias, una de ellas se basa en el balanceo de las líneas de producción. Las líneas de ensamble han sido utilizadas tradicionalmente para producir grandes cantidades de un modelo único. Este modelo es denominado línea de ensamble de un solo modelo. Por otra parte, existen los modelos de líneas de ensamble de modelo mixto, en el cual el sistema de ensamble es lo suficientemente flexible para producir más de un modelo de manera mezclada y sin patrones o tendencias; la tercera clasificación es conocida como línea de ensamble de modelos múltiples, caracterizada por producir distintos tipos de productos de una familia en una misma línea, requiriendo una configuración previa al cambio de producción (Kumar y Mahto 2013, 30), sin

embargo, su balanceo requiere técnicas diferentes de las tradicionales (Murillo-García *et a*l. 2018, 1).

Por otra parte, las líneas de ensamble pueden clasificarse de acuerdo con las restricciones del número de trabajadores por estación de trabajo: las líneas de ensamble simples, en las que por las dimensiones del producto solo es posible asignar un trabajador por estación; líneas de ensamble a dos lados, son denominadas así por presentar la factibilidad de asignar uno o dos trabajadores a cada estación de trabajo. Finalmente, las líneas de ensamble multi-tripuladas (MAL), que se caracterizan por asignar uno, dos, tres o más trabajadores por estación de trabajo (Zamzam y Elakkad 2021, 734) como se muestra en la figura 1.

Figura 1. Tipos de líneas de ensamble de acuerdo con el número de trabajadores por estación.



Fuente: Elaboración propia del modelo de Zamzam y Elakkad (2021, 734)

Una de las formas de línea de ensamble más común es conocida como multi-tripulada. Sin embargo, no ha sido explorada en su totalidad a pesar de ser común en los procesos de producción. Lo anterior genera problemas de optimización de estaciones de trabajo en líneas de ensamble; tienen sus orígenes en Talbot y Patterson (1984, 86) quienes describen un algoritmo de programación entera para asignar tareas, y determinando el mínimo número de estaciones de trabajo para balancear las estaciones. Concluyen que el algoritmo puede encontrar resultados óptimos en un tiempo computacional razonable para líneas de ensamble con máximo de 50 tareas. Adicionalmente, se han realizado propues-

tas para resolver el MAL bajo un enfoque flexible denominado FMAL (Cantos, Vidal, Sato y Magatao 2019, 1). Mediante un modelo de programación lineal entero mixto, basado en un procedimiento heurístico. Estudios adicionales consideran las habilidades de los trabajadores como factor de interés en el balanceo de líneas de ensamble (Esfandyari y Roshani 2020, 66). Por otra parte, se estudia la forma de la línea de ensamble con forma de U (Zakaraia, Zaher y Ragaa 2021, 278). Esta es también una línea de ensamble multi-tripulada.

El ALBP (*the assembly line balancing problem*) o, en español, problema de balanceo de línea de ensamble) es conocido como un problema NP-*hard* lo cual implica distribuir las tareas necesarias para fabricar cualquiera de los productos a ensamblar entre las estaciones de trabajo a lo largo de la línea de fabricación (Grzechca y Foulds 2015, 2002). Por otra parte, existen variaciones del problema de ensamble tales como SALBP (*simple assembly line balancing problem*), SALBP-1, SALBP-2 y SALBP-F (Peña-Orozco y Jiménez-Gómez 2019, 178). Algunos métodos utilizados en la solución de este tipo de problemas se centran en los heurísticos como el GRASP (*greedy randomize adaptive search procedure*) (Moreno-Ramírez 2018, 28), incluso este heurístico se utilizó en combinación con un *software* en lenguaje de programación Visual Studio 2013 Community y MS Access, para almacenar información. Se calculó el tiempo estándar en cada una de las estaciones de trabajo, para, posteriormente, balancear la línea de ensamble. Los resultados de esta investigación impactan en la reducción del tiempo ocioso en la línea de ensamble en un 84.89%. Además, se logró disminuir de 6 a 4 el número de estaciones de trabajo, logrando 33.33% de ahorro anual en dinero. También se aumentó la eficiencia de la línea de 35.24% a 72.24% (González *et al.* 2017, 1). Adicionalmente se han utilizado algoritmos heurísticos como el GRASP para abordar este problema (Paredes-Quevedo *et al.* 2022, 50).

El problema de optimización de líneas de ensamble multi-tripuladas tiene sus orígenes en la línea de ensamble de un solo modelo (Kumar y Mahto 2013, 42) propuesto por Talbot y Patterson (1984, 86) y resuelto mediante un algoritmo de programación entera. Los resultados reportan el número óptimo de estaciones de trabajo en diferentes instancias de un máximo de 111 tareas; sin embargo, el costo computacional es elevado al ser considerado también como un problema NP. El problema de líneas de ensamble multi-tripuladas es afectado por diferentes factores: el número de trabajadores en cada estación de trabajo desarrollando modelos matemáticos que lo representen con la intención de minimizar el número de trabajadores y, en una segunda instancia, minimizar el número de estaciones de trabajo mediante programación lineal (Yazdanparast y Hajihosseini 2011, 842). A pesar de propuestas de métodos exactos de programación lineal mejorados (Sato-Michels, Cantos-Lopes y Magatão 2020, 1), la complejidad computacional es una limitante en la búsqueda exhaustiva, por lo que cual se

han explorado enfoques basados en heurísticos como la optimización de colonia de hormigas (Roshani y Roshani 2012, 614), en la búsqueda de minimizar el tiempo de ciclo.

Debido a que los tiempos de las tareas dependen de la concentración de trabajadores en la estación, se presentó una formulación matemática para resolver este problema con el objetivo de minimizar el número de estaciones (Sepahi y Jalali-Naini 2014, 68); se desarrollaron cuatro procedimientos heurísticos para resolver este problema. Adicionalmente, el problema fue resuelto mediante el algoritmo de búsqueda tabú (Roshani y Giglio 2020, 194) utilizando dos mecanismos de generación de vecindarios, denominados intercambio y mutación, colaborando uno con otro de manera efectiva para encontrar nuevas soluciones factibles mediante dos listas tabú.

Por otra parte, el problema de balanceo de líneas para estaciones de trabajo multi-tripuladas fue abordado mediante algoritmos genéticos (Jithendrababu, RenjuKurian y Pradeepmon 2013, 778); demostrando mejorar la eficiencia de la línea de ensamble comparado con el sistema existente. Adicionalmente, se ha utilizado el enfoque de recocido simulado en la búsqueda por hacer más eficientes los resultados del problema (Roshani y Ghazi 2017, 34).

Otra propuesta de solución del problema de balanceo de líneas de ensamble multi-tripuladas se basa en el uso de algoritmos genéticos (Zamzam, Sadek *et al*. 2015, 59). Se propuso un nuevo indicador para evaluar el número máximo de trabajadores en la estación de trabajo, alcanzando valores óptimos en 60 instancias de 62 con mejoras en los resultados del 25% al 50%.

Al problema de balanceo de líneas de ensamble multi-tripuladas le fue adicionado un nuevo problema, el de los espacios; de tal manera que se busca minimizar el número de trabajadores en cada estación de trabajo, el número de estaciones de trabajo y el espacio o área de trabajo. Se utilizaron algoritmos genéticos (Zamzam y Elakkad 2021, 738). El MAL se ha resuelto considerando la posibilidad de que varios trabajadores realicen simultáneamente distintas tareas en un mismo puesto de trabajo. En la mayoría de los casos, se supone que los tiempos de las tareas son deterministas; toman en cuenta las posibles interferencias entre trabajadores y trata el MALBP con tiempos de trabajo en función del número de trabajadores en la estación. Se desarrollan diferentes procedimientos: resoluciones con base en un modelo matemático, dos procedimientos *relax-and-fix*, una heurística basada en la resolución de un problema de partición con restricciones (denominada "HEUR_PART") y un conjunto de otras variantes del procedimiento HEUR_PART. Los experimentos computacionales indican que HEUR_PART y la variante HEUR_PART_SGL son las propuestas que mejor funcionan (Andreu-Casas, García-Villoria, Pastor 2021, 96). El MAL se ha utilizado también para resolver problemas de manera análoga a los proble-

mas de optimización del campo electromagnético. En todos los casos considerándose un problema de alta complejidad computacional (Şahin y Kellegöz 2018, 6487).

Por lo anterior, en el presente trabajo se buscar solucionar el problema de balanceo de líneas de ensamble para estaciones de trabajo multi-tripuladas; minimizando el número de trabajadores y el número de estaciones de trabajos mediante un algoritmo genético. Comparando los resultados con los presentados en la literatura.

Presentación de la problemática

El problema de balanceo de líneas de ensamble multi-tripuladas (MALB) consiste en asignar un conjunto de tareas a un grupo de trabajadores de manera organizada y dividida mediante estaciones de trabajo, minimizando el número de estaciones de trabajo y el número de trabajadores en cada estación de trabajo. De manera formal, el modelo puede ser representado como uno de programación lineal con dos funciones objetivo:

$$FF(1) = \min(Ns) \ldots (1)$$
$$FF(2) = \min(Nm) \ldots (2)$$

La ecuación (1) busca minimizar el número de estaciones de trabajo en la línea de ensamble; la ecuación (2) es la función objetivo que busca minimizar el número de trabajadores de manera que el problema es considerado bi-objetivo.

Adicionalmente, el modelo de programación lineal tiene un conjunto de restricciones para cada una de las funciones objetivo por lo que a continuación se describen:

Función objetivo: $FF(1) = \min(Ns) \ldots (1)$
Sujeta a las restricciones:

$$\sum_{i \in W_j}^{Ns} T_i < CT \text{ para } j = 1, \ldots H \ldots (3)$$

$$Ns - 1 < Ns,\ m \in Pn \text{ para } n = 1, \ldots N \ldots (4)$$

Función objetivo: $FF(2) = \min(Nm) \ldots (2)$
Sujeta a las restricciones:

$$\Sigma_{j \in J} \Sigma_{k \in K} X_{jk} = 1 \ldots (5)$$

$$\Sigma_{j \in J} \Sigma_{k \in K} j \, x \, X_{hjk} \leq \Sigma_{j \in J} \Sigma_{k \in K} j \, x \, X_{ijk} \ldots (6)$$

La restricción (3) asegura que la suma de los tiempos de tareas asignadas a una estación en particular no exceda el tiempo de ciclo. La restricción (4) asegura que se respete la precedencia de las actividades. La restricción (5) asegura que cada tarea *i* es asignada solo a un trabajador y a una estación. La ecuación (6) es una restricción que asegura la precedencia de las actividades al asignarlas a los trabajadores.

Método de solución

Descripción general del AG

Para resolver el MALB se propone aplicar un algoritmo genético el cual primero generará un número *n* de soluciones aleatorias.

Cada solución será evaluada en la función costo la cual depende del número de estaciones, el número de trabajadores y los tiempos muertos. Una vez que cada solución es evaluada, se hará una selección de una población refinada por medio de elitismo y torneo.

Sobre la población evaluada, a cada solución se le aplicará un cruce dependiente de la posición de trabajos (JBX), el cual funciona de manera natural para problemas de secuenciación de tareas donde el orden de operaciones es crucial, pero, hasta nuestro conocimiento, no se ha aplicado todavía para este problema. Una vez realizado el cruce, se aplica una mutación en dos puntos con cierta probabilidad.

Otra parte original de este trabajo es la especificación de la función costo, la cual pondera el número de estaciones de trabajo, el número de trabajadores asignados y castiga soluciones con tiempos muertos altos, en lugar de aplicar un rebalanceo para evitar estos tiempos.

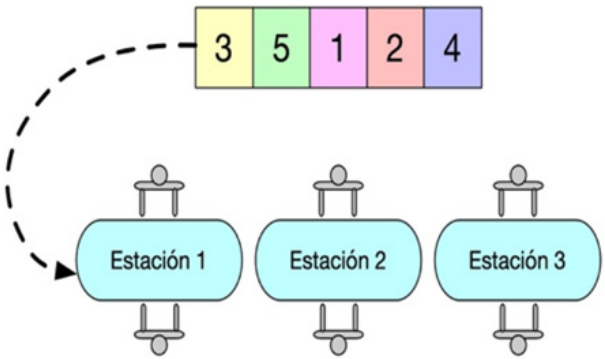
Esto hace que el AG propuesto sea más sencillo de implementar y que realmente se vea la fortaleza del proceso en descartar de manera iterativa soluciones con tiempos muertos indeseables en lugar de implementar un proceso dedicado a esta situación como se ha realizado en trabajos anteriores.

Codificación y decodificación de solución

Para una instancia del MALB con *n* trabajos por procesar, la codificación de cada solución del AG será una permutación de los *n* trabajos. Para decodificar cada solución, la cadena de *n* trabajos se leerá de izquierda a derecha, y el orden en que aparezcan los trabajos será el orden en que son procesados por el sistema. La figura 2 muestra un ejemplo de cómo una codificación de 5 trabajos se procesa en una línea multi-tripulada de 3 estaciones.

Volumen 12, número 33, (59-83), mayo–agosto 2024
DOI: https://doi.org/10.22201/ceiich.24485705e.2024.33.88239

Figura 2. Ejemplo de codificación y decodificación de una secuencia de 5 trabajos.



Fuente: Elaboración propia.

Por supuesto, se puede dar el caso en que un trabajo no pueda realizarse al no haberse cumplido las restricciones de precedencia; es decir, los trabajos precedentes del trabajo a realizar no se han procesado aún. Para este caso, se toma una pila de trabajos que no se han podido procesar, cuando ya se han tomado todos los trabajos de la solución, se revisa si la pila está vacía o no. En caso de tener trabajos pendientes, estos se van procesando con la política de primera entrada, primera salida. Si el trabajo seleccionado ya se puede procesar, se saca de la pila, en caso contrario, se toma el siguiente trabajo de la pila.

Esto se repite hasta que la pila quede vacía, con esto se asegura cumplir la precedencia de trabajos y que todos ellos sean considerados para especificar el número de estaciones, el número de trabajadores y evaluar los tiempos muertos implicados en cada solución.

Función costo y tratar tiempo muerto como castigo

Cada solución del AG será evaluada conforme al número de estaciones de trabajo N_s, al número de trabajadores N_t y al número de N_m de trabajadores que sobrepasen el umbral de tiempo muerto definido por el usuario.

Así, para una solución s_i, la función costo a minimizar por el AG está definida como:

$$f(s_i) = \alpha_1 (N_s) + \alpha_2 (N_t) + \alpha_3 (N_m)$$

Donde cada α_i es un peso que pondera cada objetivo de la función para favorecer o igualar la importancia de cada uno de ellos. Un punto relevante del trabajo es trabajar el número de trabajadores con tiempos muertos altos en la función

costo, esto simplifica el cómputo de las estaciones de trabajo y trabajadores que implica cada solución y deja el rebalanceo de la línea al AG, a diferencia de trabajos anteriores.

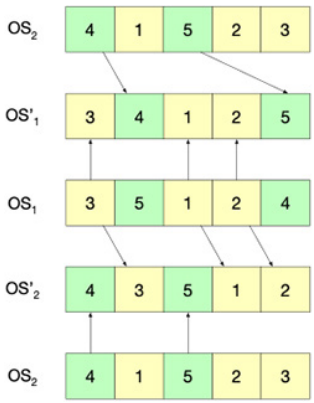
Selección, elitismo y torneo

Para la parte de selección, se utiliza primero elitismo; se toman las dos mejores soluciones (las que tenga un menor costo) como parte de la población refinada. Después, el resto de $n-2$ soluciones en la población se selecciona por medio de torneo. De la población original se eligen dos soluciones de manera aleatoria y se selecciona la que tenga un menor valor en la función costo; este proceso se repite $n-2$ veces para tener una población refinada completa.

Cruce JBX

El operador de cruce utilizado en el AG es el cruce basado en trabajos (JBX). En este cruce se definen dos subconjuntos aleatorios J_A y J_B tales que $J_A \cup J_B = J$ y $J_A \cap J_B = \emptyset$. Para dos secuencias de trabajos OS_1 y OS_2, se van a obtener dos nuevas secuencias OS'_1 y OS'_2 donde las operaciones de los trabajos J_A se colocan en OS'_1 en el mismo orden en que aparecen en OS_1, y las operaciones de los trabajos J_B se ponen en las posiciones vacías de OS'_1 guardando el orden de izquierda a derecha en que aparecen en OS_2. Se obtiene otra solución OS'_2 tomando primero las operaciones de J_B en las mismas posiciones de OS_2 y llenando los espacios vacíos de OS'_2 con las operaciones de J_A en OS_1 en el orden en que aparecen. La figura 3 presenta un ejemplo del cruce JBX para dos soluciones, cada una de 5 trabajos.

Figura 3. Ejemplo de cruce JBX para soluciones de un problema con 5 trabajos.



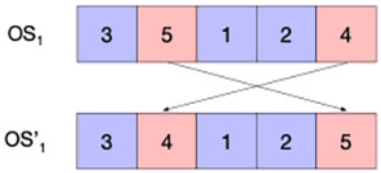
Fuente: Elaboración propia.

Volumen 12, número 33, (59-83), mayo–agosto 2024
DOI: https://doi.org/10.22201/ceiich.24485705e.2024.33.88239

Mutación

La mutación de cada solución se hace con cierta probabilidad por intercambio, en donde se seleccionan dos posiciones aleatorias de la solución seleccionada y se intercambian sus elementos para obtener una nueva solución. La figura 4 muestra un ejemplo de la mutación para una solución con 5 trabajos.

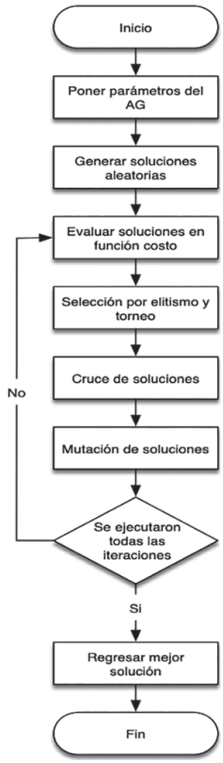
Figura 4. Ejemplo de mutación por intercambio.



Fuente: Elaboración propia.

El diagrama de flujo del AG se presenta en la figura 5.

Figura 5. Diagrama de flujo del AG.



Fuente: Elaboración propia.

Experimentación

Descripción de problemas de prueba

Para probar la efectividad del algoritmo propuesto, se tomaron 11 problemas de prueba utilizados en la literatura especializada de Zamzam, Sadek *et al*. (2015); Zamzam y Elakkad (2018) y Roshani y Giglio (2020). Cada uno de estos problemas se prueba, además, con 5 o 6 diferentes tiempos de ciclo, lo que hace un total de 64 instancias con las cuales se compara el algoritmo propuesto contra otros 6 métodos ya publicados en la literatura especializada (Dimitriadis 2006; Fattahi, Roshani y Roshani 2011; Roshani, Roshani, Roshani 2013; Zamzam y Ahmed 2021).

Características del código y máquina de prueba

El AG se implementó en Matlab R2015a (TM) en una máquina Intel Xeon W a 2.3 GHz y 128 GB de RAM. El código fuente está disponible en Github (https://github.com/juanseck/GAMmALB). Para probar la efectividad del AG se tomaron cuatro problemas con diferentes tiempos de ciclo del banco de prueba presentado en CITA.

Descripción de indicadores y pruebas estadísticas

Para comparar el rendimiento del algoritmo propuesto contra los otros métodos de solución, se tomará en cuenta el número de trabajadores (NW) y el número de estaciones de trabajo (NS) que alcance cada método en cada instancia de prueba. Mientras menores sean estos indicadores, mejor será la solución calculada.

Para el algoritmo propuesto se tomaron 30 corridas independientes por instancia, y se seleccionó la mejor solución, siguiendo la metodología empleada también por los métodos con los cuales se está comparando el algoritmo propuesto. Para los resultados de los métodos de referencias, se toman los reportados en sus respectivos artículos.

Discusión de resultados

En la tabla 1 se puede observar que el algoritmo propuesto obtiene casi siempre un resultado igual o mejor al obtenido por otros métodos de optimización. De las 64 instancias tomadas para experimentación, el algoritmo propuesto solo obtuvo un resultado peor a los obtenidos anteriormente (marcados con asterisco) y mejoró los resultados en 36 instancias, mostrando la eficacia del algoritmo para balancear la línea de producción de manera adecuada en este tipo de problemas.

Tabla 1. Comparación de los resultados obtenidos por el algoritmo propuesto contra otros publicados para los problemas de prueba.

		Zamzam y Ahmed (2021)		Roshan, Roshani, Roshani (2013)		Dimitriadis (2006)		Fattahi, Roshani y Roshani (2011)		Zamzm, Sadek, Afia y El-Kharbo (2015)		Algoritmo propuesto	
	CT	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS
Merten (7)	6	–	–	6	3	6	6	6	3	6	3	6	3
	7	–	–	5	3	5	5	5	5	5	3	5	3
	8	–	–	5	3	5	5	5	3	5	3	5	3
	10	–	–	3	3	3	3	4	3	3	3	3	2
	15	–	–	2	2	2	2	2	2	2	2	2	1
	18	–	–	2	1	2	2	2	1	2	1	2	1
Bowman (8)	17	–	–	5	5	–	–	5	5	5	5	5	3
	20	–	–	5	4	5	5	5	4	5	4	4	2
	21	–	–	5	4	–	–	5	4	5	4	4	2
	24	–	–	4	4	–	–	4	4	4	4	4	2
	28	–	–	3	2	–	–	3	2	3	2	3	2
	31	–	–	3	2	–	–	3	2	3	2	3	2
Jaeschke (9)	6	–	–	8	6	8	8	8	5	8	6	8	4
	7	–	–	7	6	7	7	7	5	7	6	7	4
	8	–	–	6	5	6	6	6	5	6	5	6	3
	10	–	–	4	4	4	4	4	4	4	4	4	2
	18	–	–	3	2	3	3	3	2	3	2	3	2
Jackson (11)	7	–	–	8	6	8	7	9	5	8	6	7	4
	9	–	–	6	4	6	5	6	4	6	4	6	3
	10	–	–	5	4	6	6	5	4	5	4	5	3
	13	–	–	4	3	4	4	4	3	4	3	4	2
	14	–	–	4	3	4	4	4	3	4	3	4	2
	21	–	–	3	2	3	3	3	2	3	2	3	2
Mansor (11)	45	–	–	5	3	–	–	5	3	5	3	5	3
	54	–	–	4	3	–	–	4	3	4	3	4	2
	63	–	–	3	2	–	–	3	2	3	2	3	2
	72	–	–	3	2	–	–	3	2	3	2	3	2
	81	–	–	3	2	–	–	3	2	3	2	3	2
Mitchell (21)	14	–	–	8	7	9	9	8	7	8	7	8	4
	15	–	–	8	7	8	8	8	7	8	7	8	4
	21	–	–	5	5	5	5	5	5	5	5	5	3
	26	–	–	5	4	5	5	5	4	5	4	5	3
	35	–	–	3	3	3	3	3	3	3	3	3	2
	39	–	–	3	2	3	3	3	2	3	2	3	2
Heskia (28)	138	8	5	8	5	8	6	8	4	8	5	8	4
	205	5	4	5	4	6	6	5	3	5	3	6	3
	216	5	3	5	3	5	4	5	3	5	3	5	3
	256	4	3	5	3	5	5	4	3	4	3	4	2
	324	4	2	4	2	4	3	4	2	4	2	4	2
	342	3	2	3	2	3	3	3	2	3	2	3	2
Kilbridge (45)	57	10	6	10	6	10	8	10	5	10	6	10	5
	79	7	4	8	4	7	6	7	5	7	5	8*	4*
	92	6	4	7	4	6	5	7	4	6	4	6	3
	110	6	3	6	3	6	5	6	3	6	3	6	3
	138	4	3	4	3	4	4	4	3	4	3	4	2
	184	3	2	3	2	3	3	3	2	3	2	3	2

Continúa ▶

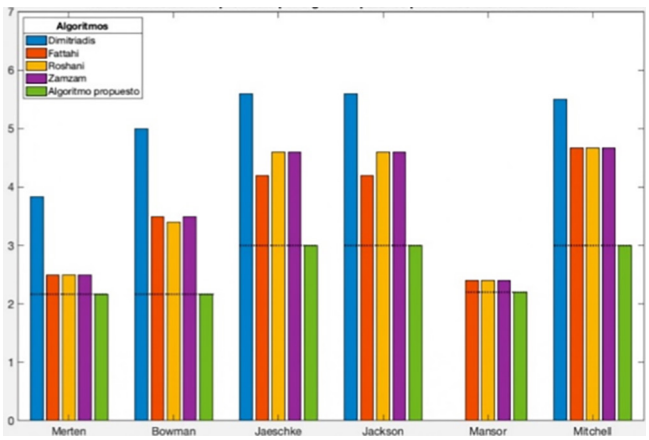
Tabla 1. Comparación de los resultados obtenidos por el algoritmo propuesto contra otros publicados para los problemas de prueba (continuación).

		Zamzam y Ahmed (2021)		Roshan, Roshani, Roshani (2013)		Dimitriadis (2006)		Fattahi, Roshani y Roshani (2011)		Zamzm, Sadek, Afia y El-Kharbo (2015)		Algoritmo propuesto	
	CT	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS
Tonge (70)	176	21	17	21	19	22	21	21	14	21	19	21	11
	364	10	6	10	7	10	9	10	5	10	7	10	5
	410	9	5	9	5	9	7	9	4	9	5	9	5
	468	8	4	8	4	8	7	8	4	8	4	8	4
	527	7	4	7	4	7	7	7	4	7	4	7	4
Arcus (83)	5048	16	11	16	11	16	16	16	11	16	11	16	8
	5853	14	10	14	10	14	13	14	10	14	9	14	7
	6842	13	8	12	8	12	10	12	8	12	8	12	6
	7571	11	9	11	10	11	11	11	7	11	7	11	6
	8412	10	8	10	8	10	10	10	6	10	6	10	5
	8998	10	5	9	7	9	8	9	6	9	6	9	5
	10816	8	6	8	5	8	8	8	6	8	5	8	4
Arcus (111)	5755	–	–	–	–	27	24	27	14	27	21	27	14
	8847	18	12	18	14	18	18	18	12	18	12	18	9
	10027	16	10	16	12	16	15	16	10	16	11	16	8
	10743	15	14	15	14	15	14	15	10	15	10	15	8
	11378	14	8	14	9	14	9	14	7	14	9	14	7
	17067	9	5	9	7	9	7	9	5	9	6	9	5

Fuente: Elaboración propia.

La figura 6 nos permite identificar el valor esperado de las estaciones de trabajo para 6 instancias, esta nos muestra que en todas las instancias el promedio de este indicador es menor a los promedios presentados por Dimitriadis (2006);

Figura 6. Número de estaciones promedio por algoritmo para los problemas Merten a Mitchell.

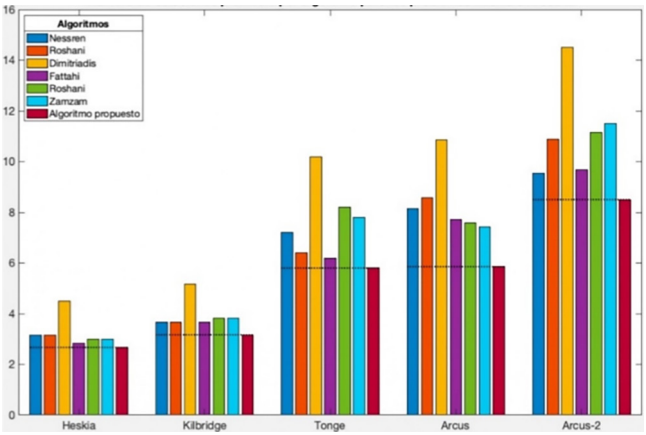


Fuente: Elaboración propia.

Fattahi, Roshani y Roshani (2011); Roshani, Roshani, Roshani (2013); Zamzam y Ahmed (2021). Los resultados son apropiados debido a que una de las funciones objetivo consiste en la minimización de las estaciones de trabajo.

Por otra parte, el número promedio de estaciones de trabajo para las instancias Heskia, Kilbridge, Tonge, Arcus y Arcus-2 es mejor para el algoritmo propuesto en este trabajo en comparación con los propuestos por Zamzam y Ahmed (2021); Roshani, Roshani, Roshani (2013); Dimitriadis (2006); Fattahi, Roshani y Roshani (2011); Roshani y Giglio (2020); Zamzam, Sadek, Afia y El-Kharbotly (2015). Como se puede observar en la figura 7 en todos los casos, el valor esperado de estaciones de trabajo es menor para el algoritmo propuesto en este documento.

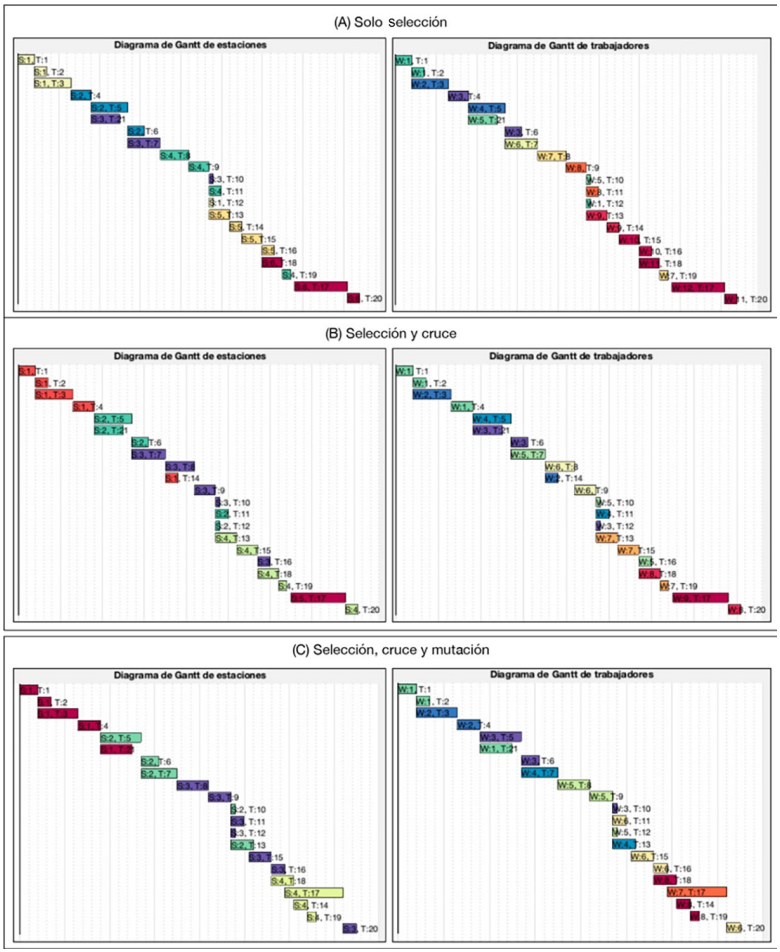
Figura 7. Número de estaciones promedio por algoritmo para los problemas Heskia a Arcus-2.



Fuente: Elaboración propia.

Como se mencionó en la sección denominada presentación de la problemática; uno de los objetivos del problema planteado consiste en asignar tareas a los trabajadores y a su vez a las estaciones de trabajo. Para representar la asignación se ilustra la figura 8, la cual contiene los diagramas de Gantt para la instancia Mitchell con un tiempo de ciclo de 14 unidades. La figura 8.A muestra los resultados de la asignación de tareas a los trabajadores para el operador genético de selección. La figura 8.B ilustra los resultados de la asignación al aplicar además de la selección el operador genético denominado cruza, evidenciando una mejora en los resultados. Finalmente, se muestran los resultados del operador genético mutación en el cual, previamente, se procesaron los operadores selección y cruza, como se puede observar en la figura 8.C. Este último operador genético tiene la intención de evitar óptimos locales.

Figura 8. Diagramas de Gantt de los trabajadores.



Fuente: Elaboración propia.

Para analizar los resultados obtenidos y compararlos respecto a la literatura, los métodos comparados son: algoritmos genéticos (Zamzam y Ahmed 2021); recocido simulado (Roshani, Roshani, Roshani 2013); heurístico de agrupación denominado GM (Dimitriadis 2006); modelo matemático y optimización de colonia de hormigas (Fattahi, Roshani y Roshani 2011); algoritmos genéticos (Zamzam, Sadek, Afia y El-Kharbotly 2015), y el algoritmo propuesto en el presente documento. Se utilizó la prueba estadística no paramétrica de Friedman; la cual se justifica debido a que el tamaño de las observaciones es menor de 10; además, no se ajusta a una distribución de probabilidad normal. En las pruebas estadísti-

cas se utilizó el *software* especializado Minitab 21.3.0. Las pruebas se realizaron para cada una de las instancias analizadas mediante el método del valor P con un nivel de significancia de 0.05 encontrando lo siguiente:

La instancia Merten (7) se probó con los algoritmos propuestos por Dimitriadis (2006), Fattahi, Roshani y Roshani (2011), Roshani, Roshani, Roshani (2013), Zamzam, Sadek, Afia y El-Kharbotly (2015) y el algoritmo propuesto en este documento. El valor P para el método ajustado para empates resultó ser de 0.003; la prueba es unilateral de orden inferior con un nivel de significancia de 0.05, por lo cual la hipótesis nula no se acepta, es decir, las medianas no son iguales como se puede observar en la figura 9. Adicionalmente, se observa que la mediana del algoritmo propuesto es menor a la de Dimitriadis (2006).

Figura 9. Resultados de la prueba de Friedman para la instancia Merten (7).

HOJA DE TRABAJO 1

Instancia Merten (7)

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	6	4.5	27.0
(Fattahi, Roshani y Roshani, 2011)	6	3.0	17.0
(Roshani, Roshani, Roshani, 2013)	6	3.0	17.0
(Zamzam, Sadek, Afia y El-Kharb, 2015)	6	3.0	17.0
Algoritmo propuesto	6	3.0	12.0
General	30	3.3	

Prueba

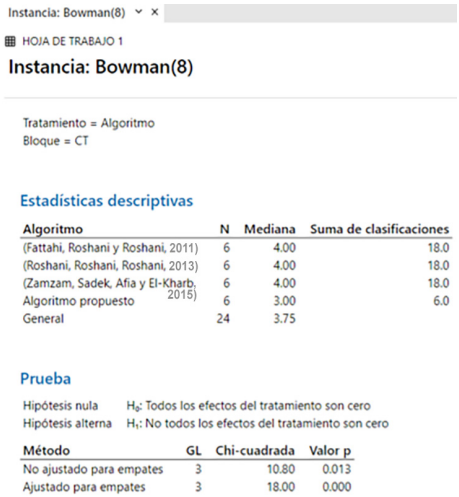
Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	4	8.00	0.092
Ajustado para empates	4	16.00	0.003

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

También se compararon los resultados de la instancia Bowman (8); la menor mediana resultó ser la del método propuesto en el presente documento, en comparación con las instancias: Dimitriadis (2006); Fattahi, Roshani y Roshani (2011); Roshani, Roshani, Roshani (2013); Zamzam, Sadek, Afia y El-Kharbotly (2015). La figura 10 muestra que el valor P obtenido por la prueba de Friedman es 0.000, con el nivel de signifiancia de 0.05 la hipótesis nula no se acepta; existe una diferencia significativa entre las medias de los métodos evaluados, y, dado que la menor media resultó ser la del algoritmo propuesto, se demuestra estadísticamente que este generó los mejores resultados.

Figura 10. Resultados de la prueba de Friedman para la instancia Bowman (8).



Instancia: Bowman(8)

HOJA DE TRABAJO 1

Instancia: Bowman(8)

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Fattahi, Roshani y Roshani, 2011)	6	4.00	18.0
(Roshani, Roshani, Roshani, 2013)	6	4.00	18.0
(Zamzam, Sadek, Afia y El-Kharb, 2015)	6	4.00	18.0
Algoritmo propuesto	6	3.00	6.0
General	24	3.75	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	3	10.80	0.013
Ajustado para empates	3	18.00	0.000

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Por otra parte, se evalúo la instancia Jaeschke (9) respecto a los métodos mostrados en la figura 11. Es posible observar que el valor P es de 0.003. Al no aceptarse la hipótesis nula, el mejor resultado es el que presenta la menor mediana, es decir, la del método propuesto con 3.2.

Figura 11. Resultados de la prueba de Friedman para la instancia Jaeschke (9).

Instancia: Jaeschke(9)

HOJA DE TRABAJO 1

Instancia: Jaeschke(9)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	5	6.0	23.5
(Fattahi, Roshani y Roshani, 2011)	5	4.8	13.0
(Roshani, Roshani, Roshani, 2013)	5	5.0	16.0
(Zamzam, Sadek, Afia y El-Kharb, 2015)	5	5.0	16.0
Algoritmo propuesto	5	3.2	6.5
General	25	4.8	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	4	12.04	0.017
Ajustado para empates	4	16.27	0.003

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

La instancia Jackson (11) se utilizó para comparar las instancias mostradas en la figura 12. Esta muestra como resultado de la prueba de hipótesis un valor P de 0.000 no aceptando a la hipótesis nula, por lo cual existe una diferencia significativa. Adicionalmente, la mediana del algoritmo propuesto es de 2.5, siendo el menor de los algoritmos comparados; esta evidencia permite concluir que es el mejor método para la instancia evaluada.

Figura 12. Resultados de la prueba de Friedman para la instancia Jackson (11).

Instancia: Jackson(11)

HOJA DE TRABAJO 1

Instancia: Jackson(11)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	6	4.5	30.0
(Fattahi, Roshani y Roshani, 2011)	6	3.5	16.5
(Roshani, Roshani, Roshani, 2013)	6	3.5	18.0
(Zamzam, Sadek, Afia y El-Kharbotly, 2015)	6	3.5	18.0
Algoritmo propuesto	6	2.5	7.5
General	30	3.5	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	4	17.10	0.002
Ajustado para empates	4	22.06	0.000

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Adicionalmente, se compararon las instancias Fattahi, Roshani y Roshani (2011); Roshani, Roshani, Roshani (2013); Zamzam, Sadek, Afia y El-Kharbotly (2015) con el algoritmo propuesto. Los resultados mostrados en la figura 13 indican que no existe una diferencia significativa entre los métodos utilizados.

La instancia Mitchell (21) fue comparada con los algoritmos mostrados en la figura 14; la hipótesis nula no se acepta en el nivel de significancia de 0.05, por lo cual las medianas son diferentes, siendo el algoritmo propuesto la menor con 3.1.

En la figura 15 se muestras muestran los resultados de la prueba de Friedman para la instancia Heskia (28), los cuales indican que el mejor resultado se presenta con el algoritmo propuesto, esto debido a que la hipótesis nula no se acepta.

En la instancia Kilbridge (45) la hipótesis nula de la prueba de Friedman no se acepta, debido a que el valor P es 0.000 y al tratarse de una prueba unilateral

Figura 13. Resultados de la prueba de Friedman para la instancia Mansor (11).

Instancia: Mansor(11)

HOJA DE TRABAJO 1

Instancia: Mansor(11)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Fattahi, Roshani y Roshani, 2011)	5	2	13.0
(Roshani, Roshani, Roshani, 2013)	5	2	13.0
(Zamzam, Sadek, Afia y El-Kharbi, 2015)	5	2	13.0
Algoritmo propuesto	5	2	11.0
General	20	2	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	3	0.36	0.948
Ajustado para empates	3	3.00	0.392

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Figura 14. Resultados de la prueba de Friedman para la instancia Mitchell (21).

HOJA DE TRABAJO 1

Instancia: Mitchell(21)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	6	5.4	27.0
(Fattahi, Roshani y Roshani, 2011)	6	4.5	18.5
(Roshani, Roshani, Roshani, 2013)	6	4.5	18.5
(Zamzam, Sadek, Afia y El-Kharbi, 2015)	6	4.5	18.5
Algoritmo propuesto	6	3.1	7.5
General	30	4.4	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	4	12.80	0.012
Ajustado para empates	4	19.69	0.001

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

de cola izquierda con un valor de 2.8 para la mediana, se observa que el algoritmo propuesto brinda el menor número de estaciones de trabajo como se observa en la figura 16.

Figura 15. Resultados de la prueba de Friedman para la instancia Heskia (28).

Instancia: Heskia(28)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	6	4.12500	36.0
(Fattahi, Roshani y Roshani, 2011)	6	2.87500	16.0
(Roshani, Roshani, Roshani, 2013)	6	3.04167	21.0
(Zamzam y Ahmed, 2021)	6	3.04167	21.0
(Zamzam, Sadek, Afia y El-Kharb, 2015)	6	2.95833	18.5
Algoritmo propuesto	6	2.70833	13.5
General	36	3.12500	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	5	14.88	0.011
Ajustado para empates	5	24.04	0.000

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Figura 16. Resultados de la prueba de Friedman para la instancia Kilbridge (45).

Instancia: Kilbridge(45) ˅ ×
HOJA DE TRABAJO 1

Instancia: Kilbridge(45)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	6	4.77083	36.0
(Fattahi, Roshani y Roshani, 2011)	6	3.52083	19.0
(Roshani, Roshani, Roshani, 2013)	6	3.43750	19.0
(Zamzam y Ahmed, 2021)	6	3.43750	19.0
(Zamzam, Sadek, Afia y El-Kharb, 2015)	6	3.60417	21.5
Algoritmo propuesto	6	2.85417	11.5
General	36	3.60417	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	5	15.60	0.008
Ajustado para empates	5	23.39	0.000

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Para la instancia Tonge (70) se obtiene un valor P de 0.001 lo que cual no permite aceptar la hipótesis nula. Esta información en conjunto con el valor de

la media de 4.6 nos permite reconocer que el método propuesto es mejor al de Dimitriadis (2006); sin embargo, no existe diferencia significativa en relación con el resto de los algoritmos de la literatura mostrados en la figura 17.

Figura 17. Resultados de la prueba de Friedman para la instancia Tonge (70).

Instancia: Tonge(70)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	5	7.33333	30.0
(Fattahi, Roshani y Roshani, 2011)	5	4.16667	10.5
(Roshani, Roshani, Roshani, 2013)	5	5.00000	18.5
(Zamzam y Ahmed, 2021)	5	4.83333	15.5
(Zamzam, Sadek, Afia y El-Kharb, 2015)	5	5.00000	18.5
Algoritmo propuesto	5	4.66667	12.0
General	30	5.16667	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	5	13.80	0.017
Ajustado para empates	5	19.80	0.001

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Para la instancia de Arcus (83) se compararon los algoritmos indicados en la figura 18. La hipótesis nula no se acepta, teniendo como menor media el algoritmo propuesto. Lo anterior implica que este método proporciona el menor número de estaciones de trabajo para esta instancia.

Finalmente, la instancia Arcus (111) se utilizó para comparar los algoritmos mostrados en la figura 19. Los resultados demuestran una diferencia significativa en las medianas; con el mínimo de estas brindadas por el algoritmo propuesto.

Conclusiones

En este trabajo se presentó un nuevo algoritmo genético para la optimización del número de trabajadores y de estaciones de trabajo en una línea de ensamble multi-tripulada. El algoritmo genético propuesto se caracteriza por castigar tiempos muertos altos en la función costo, aplicar el cruce JBX para el refinamiento de nuevas soluciones y no utilizar una política de rebalanceo fija, sino dejar este proceso al propio algoritmo genético para la generación de mejores soluciones.

Figura 18. Resultados de la prueba de Friedman para la instancia Arcus (83).

Instancia: Arcus(83)

HOJA DE TRABAJO 1

Instancia: Arcus(83)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	7	10.5833	42.0
(Fattahi, Roshani y Roshani, 2011)	7	7.7500	24.0
(Roshani, Roshani, Roshani, 2013)	7	8.2500	28.0
(Zamzam y Ahmed, 2021)	7	8.2500	25.5
(Zamzam, Sadek, Afia y El-Kharb, 2015)	7	7.0833	20.0
Algoritmo propuesto	7	5.5833	7.5
General	42	7.9167	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	5	25.67	0.000
Ajustado para empates	5	29.39	0.000

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Figura 19. Resultados de la prueba de Friedman para la instancia Arcus (111).

Instancia: Arcus (111)

HOJA DE TRABAJO 1

Instancia: Arcus (111)

Método

Tratamiento = Algoritmo
Bloque = CT

Estadísticas descriptivas

Algoritmo	N	Mediana	Suma de clasificaciones
(Dimitriadis, 2006)	5	13.4167	27.5
(Fattahi, Roshani y Roshani, 2011)	5	9.9167	11.5
(Roshani, Roshani, Roshani, 2013)	5	12.2500	25.5
(Zamzam y Ahmed, 2021)	5	10.2500	15.5
(Zamzam, Sadek, Afia y El-Kharb, 2015)	5	10.5833	18.5
Algoritmo propuesto	5	8.0833	6.5
General	30	10.7500	

Prueba

Hipótesis nula H_0: Todos los efectos del tratamiento son cero
Hipótesis alterna H_1: No todos los efectos del tratamiento son cero

Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	5	18.63	0.002
Ajustado para empates	5	21.03	0.001

Fuente: Elaboración propia utilizando el *software* Minitab 21.3.0.

Se tomaron 11 tipos de problemas de prueba y un total de 64 instancias para comparar el algoritmo propuesto con otros 6 métodos recientemente publicados, obteniendo resultados satisfactorios y mejorando en 36 de estas instancias.

En el apartado "Discusión de resultados" se pueden observar las 11 instancias evaluadas mediante la prueba de Friedman: en 10 instancias se mejoraron los resultados y en una instancia no se mejoró. Lo anterior nos permite reconocer estadísticamente que el algoritmo propuesto en el presente documento es factible y brinda mejores resultados respecto al número de estaciones de trabajo requeridas para la producción.

El algoritmo propuesto en este manuscrito es fácil de implementar, y no utiliza ninguna heurística de rebalanceo, sino que esta tarea la deja a los operadores genéticos del algoritmo, lo cual le da mayor versatilidad y disminuye el procesamiento requerido para refinar soluciones.

Por otra parte, el algoritmo maneja las variables de interés a optimizar con la linealización de una función costo, y una ponderación para cada parte de esta (número de estaciones, número de trabajadores y tiempos muertos). Un trabajo futuro propuesto es manejar estas variables utilizando técnicas multi-objetivo, basadas en frente de Pareto y dominancia de soluciones.

Otro trabajo futuro es darle más flexibilidad a la línea multi-tripulada para que un trabajador pueda desarrollar su labor en varias estaciones de trabajo contiguas. Esto requiere de la redefinición de la función costo y sus restricciones, lo cual puede ser de interés en futuras investigaciones.

Por último, se propone analizar otras metaheurísticas híbridas de optimización basadas en la inteligencia de enjambre que permitan una mayor flexibilidad entre las acciones de exploración y explotación para el cálculo de soluciones.

Referencias

Andreu-Casas, Enric, Alberto García-Villoria, Rafael Pastor. 2022. Multi-manned assembly line balancing problem with dependent task times: a heuristic based on solving a partition problem with constraints. *European Journal of Operational Research*, 96-116.

Cantos, Thiago, Giuliano Vidal, Sato Adalberto y Leandro Magatão. 2019. Flexible multi-manned assembly line balancing problem: model, heuristic procedure, and lower bounds for line length minimization. *Omega*, 1-10.

Dimitriadis, Sotirios G. 2006. Assembly line balancing and group working: a heuristic procedure for workers' groups operating on the same product and workstation. *Computers & Operations Research*, 2757-2774.

Esfandyari, Azadeh y Abdolreza, Roshani. 2020. Assembly line balancing problem with skilled and unskilled workers: the advantages of considering multi-manned workstations. *Journal of Industrial and Systems Engineering*, 66-77.

Fattahi, Parviz, Abdolreza Roshani y Abdolhassan Roshani. 2011. A mathematical model and ant colony algorithm for multi-manned assembly line balanc-

ing problem. *The International Journal of Advanced Manufacturing Technology*, 1433-3015.

González, Jaime, Katia Avilés, José Aguilar e Isaías Velázquez. 2017. *Software* para balancear línea de ensamble en la empresa Nissan Time. *Revista de Ingeniería Industrial*, 1-13.

Grzechca, W. y L. R. Foulds. 2015. The assembly line balancing problem with task splitting: a case study. *IFAC-PapersOnLine* (Elsevier), 2002-2008.

Jithendrababu, B. L., RenjuKurian y T. G. Pradeepmon. 2013. Balancing labor intensive assembly line using genetic algorithm. *International Journal of Innovative Research in Science, Engineering and Technology*, 773-779.

Kumar, Naveen y Dalgobind Mahto. 2013. Assembly line balancing: a review of developments and trends in approach to industrial application. *Global Journal of Researches in Engineering*, 28-50.

Moreno-Ramírez, Jorge. 2018. Metaheurística GRASP para el problema *vertex bisection minimization. Revista Cubana de Ciencias Informáticas*, 28-41.

Murillo-García, Raquel, Fabián Peñaherrera-Larenas, Ely Borja-Salinas y Valentino Vanegas. 2018. Líneas de ensamble y balanceo, y su impacto en la productividad de los procesos de manufactura. *Observatorio Economía Latinoamericana*, 1-19.

Paredes-Quevedo, Juan, Luis Alpala, Luis Soto-Chávez y Alberto León-Batallas. 2022. 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*, 48-57.

Peña-Orozco, Diego León y Jaime Leonardo Jiménez-Gómez. 2019. Problema de balanceo de una línea del tipo SALBP: caso de una línea de confección de prendas. *Logos, Ciencia & Tecnología*, 176-196.

Roshani, Abdolreza, Arezoo Roshani, Abdolhassan Roshani, Mohsen Salehi y Azadeh Esfandyari. 2013. A simulated annealing algorithm for multi-manned assembly line balancing problem. *Journal of Manufacturing Systems*, 238-247.

Roshani, Abdolreza y Arezoo Roshani. 2012. Multi-manned assembly line balancing problem: minimizing cycle time. *Proceedings of the IIE Asian Conference 2012*, 612-620.

Roshani, Abdolreza y Davide Giglio. 2020. A tabu search algorithm for the cost-oriented multi-manned assembly line balancing problem. *International Journal of Industrial Engineering & Production Research*, 189-202.

Roshani, Abdolreza y Farnaz Ghazi. 2017. Mixed-model multi-manned assembly line balancing problem: a mathematical model and a simulated annealing approach. *Assembly Automation*, 34-50.

Şahin, Murat y Talip Kellegöz. 2019. Balancing multi-manned assembly lines with walking workers: problem definition, mathematical formulation, and an electromagnetic field optimization algorithm. *International Journal of Production Research*, 6487- 6505.

Sato-Michels, Adalberto, Thiago Cantos-Lopes, y Leandro Magatão. 2020. An exact method with decomposition techniques and combinatorial benders' cuts for the type-2 multi-manned assembly line balancing problem. *Operations Research Perspectives*, 1-16.

Sepahi, Abdollatif y Seyed Gholamreza Jalali-Naini. 2014. Multi-manned assembly line balancing problem with variable task times. *European Journal of Academic Essays*, 68-75.

Talbot, Brian y James Patterson. 1984. An integer programming algorithm with network cuts for solving the assembly line balancing problem. *Management Science*, 85-99.

Yazdanparast, Vahid y Hamid Hajihosseini. 2011. Multi-manned production lines with labor concentration. *Australian Journal of Basic and Applied Sciences*, 839-846.

Zakaraia, Mohammad, Hegazy Zaher y Naglaa Ragaa. 2021. Stochastic local search for solving chance constrained multi-manned U-shaped assembly line balancing problem with time and space constraints. *Journal of University of Shanghai for Science and Technology*, 278-295.

Zamzam, Nessren y Ahmed Elakkad. 2021. Time and space multi-manned assembly line balancing problem using genetic algorithm. *Journal of Industrial Engineering and Management*, 733-749.

Zamzam, Nessren, Yomna Sadek, Nahid Afia y Amin El-Kharbotly. 2015. Multi-manned assembly line balancing using genetic algorithm. *International Journal of Engineering Research & Technology*, 4(12): 56-61.