UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO

INSTITUTO DE CIENCIAS BÁSICAS E INGENIERÍA

ÁREA ACADÉMICA DE INGENIERÍA Y ARQUITECTURA

Optimización de trabajadores y estaciones de trabajo en líneas de ensamble, mediante la adaptación de algoritmos genéticos.

T E S I S

QUE PARA OBTENER EL TÍTULO DE INGENIERO INDUSTRIAL

P R E S E N T A

MÁXIMO ANTONIO LÓPEZ CABRERA

Director: Dr. Gustavo Erick Anaya Fuentes

Codirector: Dr. Juan Carlos Seck Tuoh Mora

Cd. del Conocimiento, Mineral de la Reforma, Hidalgo, México.

Noviembre 2023

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE HIDALGO
Instituto de Ciencias Básicas e Ingeniería
School of Engineering and Basic Sciences

Mineral de la Reforma, Hgo., a 9 de noviembre de 2023

Número de control: ICBI-D/1612/2023
Asunto: Autorización de impresión.

MTRA. OJUKY DEL ROCÍO ISLAS MALDONADO
DIRECTORA DE ADMINISTRACIÓN ESCOLAR DE LA UAEH

Con fundamento en lo dispuesto en el Título Tercero, Capítulo I, Artículo 18 Fracción IV; Título Quinto, Capítulo II, Capítulo V, Artículo 51 Fracción IX del Estatuto General de nuestra Institución, por este medio le comunico que el Jurado asignado al Pasante de la Licenciatura en Ingeniería Industrial **Máximo Antonio López Cabrera,** quien presenta el trabajo de titulación **"Optimización de trabajadores y estaciones de trabajo en líneas de ensamble, mediante la adaptación de algoritmos genéticos"**, después de revisar el trabajo en reunión de Sinodales ha decidido autorizar la impresión del mismo, hechas las correcciones que fueron acordadas.

A continuación, firman de conformidad los integrantes del Jurado:

Presidente Dr. Norberto Hernández Romero

Secretario. Dr. Irving Barragán Vite

Vocal: Dr. Gustavo Erick Anaya Fuentes

Suplente: Dr. Juan Carlos Seck Tuoh Mora

Sin otro particular por el momento, reciba un cordial saludo.

Atentamente
"Amor, Orden y Progreso"

Dr. Otilio Arturo Acevedo Sandoval
Director del ICBI

OAAS/YCC

Ciudad del Conocimiento
Carretera Pachuca-Tulancingo Km. 4.5 Colonia
Carboneras, Mineral de la Reforma, Hidalgo,
México, C.P. 42184
Teléfono: 771 71 720 00 ext. 2231 Fax 2109
direccion_icbi@uaeh.edu.mx
www.uaeh.edu.mx

AGRADECIMIENTOS

Quisiera expresar mi más profundo agradecimiento a mi director de tesis, el Dr. Gustavo Erick Anaya y codirector el Dr. Juan Carlos Seck Tuoh. Por su orientación, experiencia, conocimiento y apoyo brindado, así como también la dedicación, comprensión, paciencia, sus valiosas y constructivas sugerencias en el desarrollo de esta investigación.

De igual manera quiero agradecer a el Dr. Joselito Medina, Dr. Norberto Hernández, Dr. Irving Barragán y al Mtro. Bernardino Martínez. Por brindar sus conocimientos, experiencias y su apoyo académico.

Gracias infinitas a mis padres y mi familia, quienes siempre me han apoyado, dándolo todo y depositado toda su confianza en mí para lograr mis objetivos académicos. Por último, pero no menos importante, a mi novia. Quien me ha amado siempre, me ha brindado su apoyo incondicional y comprensión desde los inicios de mi carrera, motivándome y recordándome que todo esfuerzo tendrá su recompensa.

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Glosario de términos.

Heurístico: es un procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución. (Díaz, 1996).

Metaheurísticos: son métodos aproximados diseñados para resolver problemas de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos, combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos (Ibrahim H & James P, 1996).

Elitismo: pretende asegurar que aquel o aquellos individuos que son los más aptos de la población actual sobrevivan y continúen participando en el proceso evolutivo, pasando a la siguiente generación de manera intacta, sin recombinarse ni mutarse. (Aspiazu, 2010).

Mutación: es una alteración en la información hereditaria. Las mutaciones pueden tener efecto positivo, nulo o negativo; y pueden causar enfermedad. (Feliciano, 2020).

Resumen

El problema de balanceo de líneas de ensamble se presenta de manera frecuente en las organizaciones dedicadas a procesos de transformación y de oferta de servicios; los objetivos son la minimización del número de trabajadores y estaciones de trabajo para minimizar los costos de producción. El problema es clasificado como NP completo debido a la complejidad computacional generada por el crecimiento exponencial de soluciones conforme crece el número de tareas en el proceso. Adicionalmente el problema se vuelve más complejo cuando por el tamaño del producto los operadores pueden realizar tareas simultáneas en cada estación de trabajo. Este es conocido como problema multi-tripulado para el balanceo de línea de ensamble tipo dos; debido a que se busca minimizar el número de estaciones de trabajo y el número de operadores dado un tiempo de ciclo y sin un límite de estaciones de trabajo.

La presente tesis busca solucionar al problema de manera multi-objetiva, considerando la minimización de las estaciones de trabajo, así como el número de trabajadores; se propone el uso de un algoritmo genético (AG) que en nuestro conocimiento no se ha utilizado para este problema: multi-tripulado y multi-objetivo. El método propuesto se codificó de manera tal que las soluciones aleatorias en los operadores genéticos son todas factibles lo que permite mejorar la eficiencia del algoritmo en lugar de generar soluciones aleatorias y después recolectar a las factibles, esta es otra de las innovaciones de esta tesis.

Los resultados demuestran que es posible reducir el número de estaciones de trabajo y trabajadores, asignándolos de manera balanceada, sin embargo, al tratarse de un problema bi-objetivo, además de utilizar algoritmos heurísticos, los

resultados no aseguran valores óptimos, pero presentan buenas soluciones en un tiempo computacional adecuado como la industria lo requiere.

Abstract

The problem of balancing assembly lines occurs frequently in organizations dedicated to transformation processes and service offerings; The objectives is to find the minimum number of workers and workstations to reduce production costs. The problem is classified as NP-complete due to the computational complexity caused by the exponential growth of solutions as the number of tasks in the process increases. Additionally, the problem becomes more complex when, due to the size of the product, operators can perform simultaneous tasks at each workstation. This is known as the multi-manned problem for type two assembly line balancing, because the aim is to minimize the number of workstations and the number of operators given a cycle time and without a limit of workstations.

This thesis seeks to solve the problem in a multi-objective way, considering the minimization of workstations, as well as the number of workers, The use of a genetic algorithm that, to our knowledge, has not been used for this problem is proposed: multimanned and multi-objective. The proposed method was coded in such a way that the random solutions in the genetic operators are all feasible, which allows improving the efficiency of the algorithm instead of generating random solutions and then collecting the feasible ones, this is another of the innovations of this thesis.

The results show that it is possible to reduce the number of workstations and workers, assigning them in a balanced manner, however, since it is a bi-objective problem, in addition to using heuristic algorithms, the results do not ensure optimal values, but present good solutions in adequate computing time as the industry requires.

Introducción

La competitividad que se generó debido a la globalización obliga a las organizaciones con fines de lucro a buscar mecanismos y estrategias que les mantengan en la participación del mercado, mediante indicadores sujetos a la mejora continua, tales como la calidad, producción, tiempos y costos los cuales se vinculan y pueden asociarse a un indicador único, la productividad, entendida como el cociente de lo producido y los requerimientos. En este sentido, la minimización de los costos de producción es un objetivo primordial para los directivos de las organizaciones. 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 históricamente para producir grandes volúmenes de un modelo único, este modelo es denominado como Línea de ensamble de un solo modelo, por otra parte existen los modelos denominados, línea 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 y 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, 2013). Sin embargo, su balanceo requiere técnicas diferentes de las tradicionales (Raquel Murillo Garcia, 2018).

Por otra parte, las líneas de ensamble pueden clasificarse de acuerdo a 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 a un trabajador por estación; las 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 N. y., 2021). El presente trabajo

de investigación propone un algoritmo eficiente que optimice los costos mediante la minimización de las estaciones de trabajo y el número de trabajadores, para el problema de balanceo de líneas de ensamble con estaciones de trabajo multi-tripuladas.

Capítulo 1. Antecedentes, propósito y organización

1.1. Antecedentes

La revolución industrial surgió en inicios del siglo XVII hasta finales del siglo XX, en este periodo comenzó la producción en serie o en cadena con lo cual se pretendía tener una organización de producción en donde el trabajador esté especializado en una sola función para que éste elevará la calidad y los tiempos de producción. Sin embargo, a inicios del siglo XX surge el Taylorismo el cual está basado en la división del trabajo, la producción en cadena y la eliminación de la autonomía temporal del Trabajador**,** (López, 2020)**.** Con esto se logró reducir la forma efectiva de los costos de fabricación por lo cual se bajó la cantidad de trabajadores causando muchos problemas. Poco después surge el Fordismo donde Henrry Ford fundador de Ford Motor Company fue el impulsor en la práctica de producción en serie, línea de montaje y eliminó la flexibilidad de los trabajadores. Con esto se implementó la producción del Ford T con una combinación y organización general del trabajo altamente especializada y reglamentada a través de cadenas de montaje, maquinaria especializada, salarios más elevados y un número elevado de trabajadores en plantilla (Andrea, 2019).

Hoy en día tradicionalmente cuando se llegan a producir grandes cantidades de un modelo único se utilizan las líneas de ensamble. Sin embargo, es fácil producir ambientes de manufactura flexible con lo cual se implementa una línea de modelos mixtos. (Raquel Murillo Garcia, 2018). La competitividad en el sector industrial nacional e internacional exige variabilidad de productos, donde la flexibilidad de los sistemas productivos es el eje principal. Sin embargo, el manejo inadecuado de los recursos disminuye el factor de

competencia en los mercados, pero uno de los recursos más importantes en una empresa es el tiempo; su inadecuada utilización ocasiona retrasos tanto en la producción como en la entrega de un bien o servicio. También la falta de una clara asignación en las actividades de acuerdo con el tiempo disponible puede generar tiempo ocioso. (González-Jaime, 2017)**.**

Así como también se plantean algoritmos metaheurísticos para el problema de programación de producción en un taller de flujo, considerando como un problema no polinómico completo debido a su dificultad. Por lo tanto, el estudio de la problemática es relevante dada a la utilidad en la práctica, es por eso por lo que en las fábricas con unas líneas de ensamble o también en la planificación de cadenas de suministro colaborativo, por lo tanto, se investigó acerca del estudio para evaluar los algoritmos metaheurísticos y un algoritmo genético (Paredes-Quevedo, Alpala, Soto-Chávez, & León-Batallas, 2022).

Otro ejemplo se presenta en la producción de baterías de plomo-ácido son generalmente utilizadas en vehículos convencionales y eléctricos el cual su proceso de fabricación es caracterizado por tener una larga serie de actividades rutinarias. Con esto se diseñó una propuesta para la mejora utilizando el balanceo de línea con el propósito de incrementar la productividad en la fabricación de pilas expandidas en ese tipo de materias y de esa forma podrá equiparar cargas de trabajo en las estaciones del proceso, mediante la cual se minimizará el tiempo ocio y logrando la optimización del tiempo de producción (Riquett Rodríguez, 2021)**.**

A pesar del creciente nivel de automatización industrial, el ensamble manual continúa desempeñando un rol fundamental en diversos sectores de la manufactura. Por lo tanto, las operaciones de tipo manual son susceptibles de los errores que son cometidos por los humanos la cual ocasiona problemas que afectan la calidad y las pérdidas económicas. Se propuso mostrar ciertos

métodos los cuales permiten identificar diferentes tipos de errores y evaluar la influencia de factores los cuales afectan el desempeño del trabajador, (Torres-Medina , 2020)**.**

1.2. Planteamiento del problema

El problema Multi-Manned Assembly Line Balancing (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 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:

$$FF_1 = \min\,(N_s) \quad (1)$$

$$FF_2 = \min\,(N_m) \quad (2)$$

La ecuación (1) busca minimizar el número de estaciones de trabajo en la línea de ensamble, en tanto, 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:

Sujeta a las restricciones

$$\sum TiNsi \in Wj < CT \quad for\ j = 1, \dots H \dots (3)$$

$$Ns - 1 < Ns, m \in Pn \quad for\ n = 1, \dots N \dots (4)$$

Función objetivo: $FF(2) = \min(Nm) \ldots (2)$

Sujeta a las restricciones

$$\sum j \in j \sum k \in KX\, jk = 1 \ldots . (5)$$

$$\sum j \in j \sum k \in K\, j\, x\, Xhjk \leq \sum j \in j \sum k \in k\, j\, x\, Xijk \ldots (6)$$

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.

1.3. Propósito de la investigación

La presente investigación tiene como propósito formular un algoritmo computacional eficiente que defina secuencias de trabajos para el Problema de líneas de ensamble para estaciones de trabajo multi-tripuladas, que minimice el número de estaciones de trabajo, así como el número de trabajadores en cada una de ellas, considerando tiempos de ciclo y actividades precedentes de manera predeterminada.

1.4. Objetivo general

Diseñar un modelo de asignación de actividades para el problema de balanceo de líneas de ensamble multi-tripuladas, mediante un algoritmo genético, para minimizar el número de trabajadores y estaciones de trabajo.

1.5. Objetivos específicos.

- Reconocer el modelo de programación lineal que resuelva el problema de balanceo de líneas de ensamble multi-tripuladas.
- Adaptar al heurístico denominado algoritmos genéticos para resolver el problema de balanceo de líneas de ensamble multi-tripuladas.
- Evaluar la eficiencia del modelo heurístico propuesto mediante experimentación computacional.
- Comparar los resultados obtenidos mediante el algoritmo propuesto, respecto a los de la literatura.

1.6. Justificación de la investigación

La industria de la transformación presenta grandes retos a nivel mundial. Actualmente, uno de los problemas que con mayor frecuencia se presenta es el balanceo de líneas de ensamble. Este es un problema conocido como NP-Hard, debido a que al aumentar el número de tareas a asignar a las estaciones de trabajo el cual es uno de los objetivos del problema en cuestión, el número de combinaciones crece exponencialmente generando dificultades computacionales para realizar una búsqueda exhaustiva. Es por lo anterior, que las investigaciones se han enfocado a utilizar heurísticos para su solución, evaluando la eficiencia mediante parámetros como: la proximidad a los mejores resultados encontrados en la literatura, mejores tiempos computacionales reportados, entre otros.

El problema se vuelve más complejo cuando debido al tamaño (relativamente grande) del producto a maquinar permite tareas simultaneas en la estación de trabajo; a esto se le denomina estación multi-tripulada.

Esta investigación tiene como enfoque proponer, evaluar mediante la comparación con instancias de la literatura mediante una propuesta de Algoritmo Genético, el cual genera soluciones aleatorias factibles evitando pasos de depuración posteriores, lo cual en caso de ser exitoso beneficiará a toda organización que en sus procesos planifique su producción cuando esta se realiza en una línea de ensamble con estaciones de trabajo multi-tripuladas.

1.7. Alcance.

El alcance de esta investigación radica en la comparación de la eficiencia del algoritmo propuesto frente a métodos y técnicas presentadas en la literatura.

Capítulo 2. Marco teórico.

2.1. Balanceo de líneas de ensamble.

El origen de las líneas de ensamble se remonta al año de 1913 en la planta Highland Park de Henry Ford donde fue introducida por primera vez, esto ocasionó una revolución automotriz y el concepto de la fabricación en todo el mundo. En consecuencia, tuvo una gran incidencia en el desarrollo de la Segunda Revolución Industrial.

Una línea de ensamble manual consiste en múltiples estaciones de trabajo ordenadas en forma secuencial en las cuales trabajadores humanos ejecutan operaciones de ensamble (Groover, 2007).

El procedimiento usual en una línea manual empieza con el lanzamiento de una pieza base en el extremo inicial de la línea. Con frecuencia se requiere un transportador de trabajo que contenga la pieza durante su movimiento a lo largo de la línea. La pieza base viaja por cada una de las estaciones, donde los trabajadores realizan tareas que construyen el producto de forma progresiva. En casa estación de agregan componentes a la pieza base hasta que todo el contenido de trabajo se ha terminado cuando el producto sale de la estación final. (Groover, 2007)

En 1984 se presentó un artículo, en el cual se describió un algoritmo de programación entera para asignar las tintas en una línea de montaje a las estaciones de trabajo de tal manera que el número de estaciones de trabajo sea mínimo para la tasa de producción deseada. En el desarrollo se logró encontrar el procedimiento donde se supuso que los nodos de la red son numerados de tal manera que, si la tarea m precede a la tarea n, $m<n$. Con eso se diagnosticaron muchos intentos para resolver ese problema con el uso

de la programación entera. Por lo tanto, el enfoque se basa en la enumeración implícita de balas que reduce los requisitos de almacenamiento del ordenador con respecto a otro método. El objetivo es minimizar el número de puestos de trabajo, que es la estación a la que esta asignada la única tarea terminal, las restricciones aseguran que la suma de los tiempos de las tareas asignadas a una demanda estación no supere los tiempos de ciclo. Las restricciones también aseguran que todos los predecesores tecnológicos de una determinada tarea sean realizados antes de llevarla a cabo.

Conceptualmente en el problema se asignaron algunas restricciones, las cuales fueron; (I) asegura que la suma de los tiempos de las tareas asignadas en una estación no exceda el tiempo de ciclo y (II) asegura que todos los predecesores tecnológicos de una tarea determinada se realicen antes de la siguiente.

El principal obstáculo para aplicar directamente esta formulación es la dificultad de especiación operativa (I). Intentos anteriores de programación entera (Petterson, 1975), (Thangavelu, 1971) y (W.W, 1961) (Browman, 1960) han llevado a las formulaciones que utilizan variables 0-1 que sí permiten la representación explícita de restricciones de ciclo, ocurrencia y precedencia. La consecuencia de utilizar variables 0-1**,** sin embargo, la formulación resultante contiene un gran número de variables y restricciones. Esto se demuestra en (Petterson, 1975). Las formulaciones del problema ALB se comparan mediante un ejemplo de ocho tareas problemáticas.

A continuación, se definen algunas variables a considerar en el problema:

El tiempo de ciclo (Tc): se puede definir como el tiempo que transcurre entre la producción de dos unidades consecutivas de un producto. Se expresa en: segundos/unidad, o minutos/unidad. Representa la cadencia o rapidez del

proceso. El tiempo de ciclo tiene suma importancia ya que las líneas de ensamble es un proceso cíclico y determina la capacidad que tiene la línea. La capacidad de la línea es la inversa del tiempo de ciclo (unidades/tiempo)

El tiempo de proceso (Tp): es el tiempo que se emplea en producir una única unidad desde el principio al fin de la línea. Representa la suma de los tiempos parciales de cada uno de los subprocesos, sin contar los tiempos de espera debido al stock intermedio.

El tiempo de flujo (Tf): es el tiempo que transcurre el componente de producto que tenga un recorrido mayor al atravesar todo el proceso productivo desde principio a fin. Representa la suma del tiempo de proceso más el tiempo debido a las demoras en los stocks intermedios. Este valor da idea de la agilidad del proceso, o de la capacidad de reaccionar ante un cambio requerido por un cliente.

El takt time: Esta es una definición de tiempo que suele confundirse con el tiempo de ciclo. El takt time relaciona la demanda de los clientes con la disponibilidad de tiempo productivo. Lo hace midiendo el ritmo al cual deberíamos producir para satisfacer la demanda del cliente de forma exacta (producción ajustada). (Cristofani, 2023)

El Balanceo de Líneas de Ensamble consiste en conjuntar actividades u operaciones que cumplan con un tiempo de producción para que no exista cuellos de botella en una línea de producción.

La idea principal de una línea de ensamble se trata de que un producto se arma gradualmente a medida que es transportado, pasando frente a

estaciones de trabajo relativamente fijas, por un dispositivo de manejo de materiales.

El problema de balanceo de línea de ensamble (ALBP): consiste en hallar un balance de línea (asignación de cada tarea a una estación) tal que se cumplan restricciones de precedencia, entre otras restricciones. (Pena-Orozco, 2019). Los problemas de equilibrio de la línea de montaje (ALBP) surgen cada vez de que se configura, rediseña o ajusta una línea de montaje. Un ALBP consiste en distribuir la carga de trabajo total para fabricar cualquier unidad de los productos a ensamblar entre las estaciones de trabajo a lo largo de la línea.

El problema de equilibrio de la línea de montaje dependiente de la secuencia (SDALBP) es una extensión del problema de equilibrio de la línea de montaje simple estándar (SALBP) que tiene una relevancia significativa en la configuración de la línea de montaje del mundo real. SDALBP amplía el problema básico al considerar tiempos de tareas dependientes de la secuencia**.**

En el artículo de investigación de (Scholl A, 2008) se define el nuevo problema con varias versiones de un programa de enteros mixtos, adaptan enfoques de solución para SALBP a SDALBP, generan datos de prueba y realizan algunos experimentos computacionales preliminares.

La minimización de los costos de producción es un objetivo primordial para los directivos de las organizaciones por lo tanto 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 como Línea de ensamble de un solo modelo, por otra parte existen los

modelos denominados línea 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 y 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).

Los problemas de equilibrio de la línea de montaje (ALBP) surgen cada vez de que se configura, rediseña o ajusta una línea de montaje. Un ALBP consiste en distribuir la carga de trabajo total para fabricar cualquier unidad de los productos a ensamblar entre las estaciones de trabajo a lo largo de la línea. El problema de equilibrio de la línea de montaje dependiente de la secuencia (SDALBP) es una extensión del problema de equilibrio de la línea de montaje simple estándar (SALBP) que tiene una relevancia significativa en la configuración de la línea de montaje del mundo real. SDALBP amplía el problema básico al considerar tiempos de tareas dependientes de la secuencia**.** En este trabajo de investigación de investigación de (Scholl A, 2008) se define el nuevo problema con varias versiones de un programa de enteros mixtos, adaptan enfoques de solución para SALBP a SDALBP, generan datos de prueba y realizan algunos experimentos computacionales preliminares.

La minimización de los costos de producción es un objetivo primordial para los directivos de las organizaciones. 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 como Línea de ensamble de un solo modelo, por otra parte existen los modelos denominados línea 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 y 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). Sin embargo, su balanceo requiere técnicas diferentes de las tradicionales (Raquel Murillo Garcia, 2018).

Las líneas de ensamble se clasifican de acuerdo al número de trabajadores por estación de trabajo, en primer lugar se encuentra a las Líneas de ensamble simples, en las que por las dimensiones del producto solo es posible asignar a un trabajador por estación; por otra parte están las 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 se encuentran las líneas de ensamble multi-tripuladas (MAL), que se caracterizan por asignar uno, dos, tres o más trabajadores por estación de trabajo (Zamzam N. y., 2021).

El problema de optimización de líneas de ensamble multi-tripuladas tiene sus orígenes en la línea de ensamble de un solo modelo (Kumar, 2013) propuesto por (Talbot, 1984) y resuelto mediante un algoritmo de programación entera, reportando 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, uno de ellos es 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, 2011). A pesar de propuestas de métodos exactos de programación lineal mejorados (Sato-

Michels, 2020), la complejidad computacional es una limitante en la búsqueda exhaustiva, por lo que se han explorado enfoques basados en heurísticos como la optimización de colonia de hormigas (Roshani A. y., 2012), en la búsqueda de minimizar el tiempo de ciclo.

A continuación, se describen algunos métodos utilizados para el balanceo de líneas de ensamble o de producción:

2.2. Método búsqueda exhaustiva

La búsqueda exhaustiva es una técnica general de resolución de problemas en la que se realiza una búsqueda exhaustiva y sistemática en el espacio de soluciones. Sin embargo, suele resultar ineficiente cuando el espacio muestral no puede resolverse en un tiempo polinómico convirtiéndose en un problema NP-hard. La búsqueda se suele realizar recorriendo un árbol con el que se representan las posibles soluciones.

Se trata de un tipo particular de algoritmos de fuerza bruta que se emplean para problemas donde se busca un elemento con una propiedad especial, usualmente entre objetos combinatorios como permutaciones, combinaciones, o subconjuntos.

El método general consiste en:

1. Generar una lista de todas las soluciones potenciales del problema en una forma sistemática.
2. Evaluar las soluciones potenciales una a una, descalificando las no factibles y manteniendo un registro de la mejor encontrada hasta el momento (en problemas de optimización).
3. Cuando la búsqueda finalice, regresar la mejor solución encontrada (Tello, 2018).

Hay algunos métodos de recorrido del árbol:

– Recorrido en anchura.

– Backtracking.

– Branch and Bound (ramificación y acotación).

2.3. Método de ramificación y acotación (Branch and Bound)

El método de ramificación y acotación o también llamado Branch and Bound, resuelve el problema de tal forma que, si la solución a este verifica condiciones de integridad, entonces también es la solución al problema entero, de lo contrario se comienza con la ramificación del problema. La ramificación consiste en dividir cada problema en dos nuevos subproblemas, obtenidos mediante el uso de restricciones excluyentes que dividen el conjunto de oportunidades del problema original en dos partes, pero eliminando en ambas partes la solución no entera del problema original. Cuando en la solución al problema una variable que es entera xi toma el valor xbi no entero, entonces se generan, a partir de dicho valor, dos restricciones $xi \leq [xbi]$ y $xi \geq [xbi] + 1$ siendo $[xbi]$ la parte entera por defecto de (xbi).

2.4. Método (algoritmo) aditivo de balas

El método aditivo de Egon Balas, hace referencia a un procedimiento de enumeración que encuentra el óptimo de manera más eficiente, evaluando algunas soluciones. Se deben poner todas las variables iguales a cero y luego asignar a una por una de las variables el valor 1 (Hillier & Lieberman, 2010).

Posteriormente se reemplaza en cada una de las restricciones y se averigua la infactibilidad:

Paso 1. La función objetivo debe ser del tipo minimización, con todos los coeficientes no negativos.

Paso 2. Todas las restricciones deben ser de tipo entero con los lados derechos negativos de ser necesario. Después estas restricciones se convierten a ecuaciones, usando las variables auxiliares en el lado izquierdo de las restricciones.

2.5. Ramificación y poda

Una búsqueda en anchura recorre los nodos de un grafo por niveles a partir de un nodo raíz: en primer lugar, los nodos conectados directamente con el nodo raíz (nivel 1); a continuación, los nodos conectados con el nodo raíz a través de un único nodo intermedio (nivel 2), etc.

La estrategia de ramificación y poda generaliza la búsqueda en anchura manteniendo una estructura de nodos vivos (esto es, nodos creados, pero no expandidos aun, que pueden estar en distintos niveles del árbol) y recorriéndola en un cierto orden inteligente (que no es ni en profundidad ni en anchura).

Para ello, cada nodo vivo v es etiquetado con una estimación heurística de la probabilidad de encontrar una solución a partir de la solución parcial que dicho nodo representa. En problemas de optimización, se asocia a cada nodo una cota optimista del valor máximo que la función objetivo puede alcanzar al expandir dicho nodo: $fmax(v)$.

Los nodos se ordenan entonces de acuerdo con dicho valor, lo que permite explorar en primer lugar, los nodos más prometedores. Si lo que se busca es una solución cualquiera, esta estrategia reduce el número de nodos que es

necesario expandir para encontrarla y, por tanto, el coste de la búsqueda (Taha, 2012).

En problemas de optimización, a medida que van encontrándose soluciones completas y se dispone del valor de la función objetivo para la solución óptima, f (spot), es posible eliminar de la lista de nodos vivos (esto es, podar) aquellos nodos para los que $fmax\ (v) \leq f\ (spot)$.

Como parte de los Algoritmos Genéticos son fundamentados como último paso antes de obtener la mejor solución, se aplicará un Algoritmo Elitista o de Búsqueda Heurística, con el propósito de obtener el mejor resultado.

Naturalmente se basa en la aplicación de algún método existente que se enfoque en la optimización, con esto se busca que se presente el menor factor de rechazo, por lo cual se ve en la necesidad de utilizar restricciones buscando la solución de cara.

2.5. Estrategia búsqueda exhaustiva por construcción de soluciones

La búsqueda exhaustiva por construcción de soluciones es una estrategia que tiene como objetivo la construcción de respuestas al problema mediante el desarrollo de procedimientos específicos que dependen de cada situación. La ejecución de esta estrategia generalmente permite establecer no solo una respuesta, sino que permite visualizar la globalidad de soluciones que se ajustan al problema.

2.6. Programación entera binaria

La programación entera binaria es un método perteneciente a la programación lineal, por lo tanto, es formulado a través de ecuaciones lineales logrando optimizar así una función objetivo lineal. También la programación binaria es utilizada en problemas de asignación en donde se enfoca en hacer o no una tarea.

Metodología de solución de un pequeño problema simple de balanceo de línea de ensamble. La Tabla 1 muestra en resumen las características de los problemas SALBP`s; mientras tanto, la Tabla 2 proporciona los tiempos de tareas y las actividades de precedencia, las cuales son restricciones al problema en cuestión. Adicionalmente, el SALBP contiene un tiempo de ciclo de 100 segundos para un caso particular, sin embargo, este pude variar dependo de las características de cada caso. Finalmente, la Figura 1 representa las restricciones de precedencia mencionadas en la Tabla 2. Por ejemplo, para este caso las actividades B, C no pueden realizarse hasta que no se concluya A, mientras que ésta no tiene restricción de precedencia.

Tabla 1. Clasificación de características SALBP'S

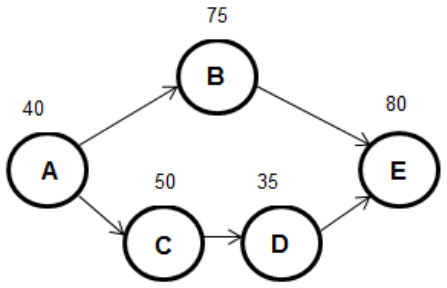
Versión	Tiempo de Ciclo	Números de Estaciones
SALBP-F	Fijo	Fijo
SALBP-1	Fijo	Mínimo
SALBP-2	Mínimo	Fijo
SALBP-E	Mínimo	Mínimo

Tabla 2. Tiempos de espera

Tarea	Tiempo	Predecesor in
A	40	-
B	75	A
C	50	A
D	35	C
E	80	B, D

En la Figura 1 explica se muestra el diagrama del flujo y restricciones de procedencia; por ejemplo, es necesario realizar la tarea A previa la B; la tarea D tiene como precedente a la C y ésta a su vez a la A.

Figura 1. Diagrama de precedencia



Fuente: Elaboración propia a partir de (Escobar Alvarán, Garcés Hincapié, & Restrepo Correa, 2012)

Solución

Identificamos el problema como un SALBP, tipo 1, ya que el tiempo de ciclo es dado, (100 segundos), se va a emplear la programación entera binaria para encontrar el mínimo número de máquinas o estaciones de la línea.

Se plantea la función objetivo de acuerdo con el siguiente criterio.

$$Min\ Z = \sum_{i=1}^{N} \sum_{k=1}^{k} C_{ik} X_{ik}$$

Donde

Ci = coeficiente de costo

Donde

estación $k; \acute{o}\ Xik = 0$, de lo contrario.

Ci = coeficiente de $CostoXik = 1$, si la tarea i es asignada a la o Se plantean las restricciones de problema así́:

Restricción de tiempo de ciclo: $\sum_{i=1}^{n} Ti\ Xik \leq C$ para k= 1, 2, …, k.

Restricción asignación unitaria: $\sum_{\mathrm{k}=1}^{\mathrm{k}} \mathrm{Xik} = 1$ para i =1, 2, …, N

Restricción de precedencia: $Xvb \leq \sum_{j=1}^{b} Xuj$ para b= 1, 2, ..., k y (u, v) ε IP: Restricción de precedencia.

Inicialización:

Xik = Tarea i asignada a la estación k, donde i = a, b, c, d, e; Y k = 1, 2, 3, 4, 5.

Planteamos la función objetivo en donde añadimos pesos posicionales de 1, 2, 3, 4 y 5 a cada máquina respectivamente, con el fin de que la máquina o estación uno tenga prioridad sobre la 2, 3, 4 y 5, la 2 sobre la 3, 4 y 5; y así respectivamente, tal que si no es necesario el implemento de una estación de trabajo se ocupen las primeras estaciones prioritariamente.

$$X_{a1} + 2X_{a2} + 3X_{a3} + 4X_{a4} + 5X_{a5} + X_{b1} + 2X_{b2} + 3X_{b3} + 4X_{b4} + 5X_{b5} + X_{c1} + 2X_{c2} + 3X_{c3} + 4X_{c4} + 5X_{c5} + X_{d1} + 2X_{d2} + 3X_{d3} + 4X_{d4} + 5X_{d5} + X_{e1} + 2X_{e2} + 3X_{e3} + 4X_{e4} + 5X_{e5}$$

Ahora planteamos las restricciones empezando por la restricción del tiempo de ciclo:

1. $40X_{a1} + 75X_{b1} + 50X_{c1} + 35X_{d1} + 80X_{e1} \leq 100$
2. $40X_{a2} + 75X_{b2} + 50X_{c2} + 35X_{d2} + 80X_{e2} \leq 100$
3. $40X_{a3} + 75X_{b3} + 50X_{c3} + 35X_{d3} + 80X_{e3} \leq 100$
4. $40X_{a4} + 75X_{b4} + 50X_{c4} + 35X_{d4} + 80X_{e4} \leq 100$
5. $40X_{a5} + 75X_{b5} + 50X_{c5} + 35X_{d5} + 80X_{e5} \leq 100$

A continuación, se plantean las restricciones de asignación

6. $X_{a1} + X_{a2} + X_{a3} + X_{a4} + X_{e5} \leq 1$

7. $X_{b1} + X_{b2} + X_{b3} + X_{b4} + X_{b5} \leq 1$

8. $X_{c1} + X_{c2} + X_{c3} + X_{c4} + X_{c5} \leq 1$

9. $X_{d1} + X_{d2} + X_{d3} + X_{d4} + X_{d5} \leq 1$

10. $X_{e1} + X_{e2} + X_{e3} + X_{e4} + X_{e5} \leq 1$

Y por último planteamos las restricciones de precedencia, empezando con las correspondientes a la restricción de que el predecesor inmediato de b es a, así que ha debe estar asignado para poder asignar b.

(11) $X_{b1} \leq X_{a1}$

(12) $X_{b2} \leq X_{a1} + X_{a2}$

(13) $X_{b3} \leq X_{a1} + X_{a2} + X_{a3}$

(14) $X_{b4} \leq X_{a1} + X_{a2} + X_{a3} + X_{a4}$

(15) $X_{b5} \leq X_{a1} + X_{a2} + X_{a3} + X_{a4} + X_{a5}$

El predecesor inmediato de c es a, cumpliéndose por lo tanto lo siguiente

(16) $X_{c1} \leq X_{a1}$

(17) $X_{c2} \leq X_{a1} + X_{a2}$

(18) $X_{c3} \leq X_{a1} + X_{a2} + X_{a3}$

(19) $X_{c4} \leq X_{a1} + X_{a2} + X_{a3} + X_{a4}$

(20) $X_{c5} \leq X_{a1} + X_{a2} + X_{a3} + X_{a4} + X_{a5}$

El predecesor inmediato de d es c:

(21) $X_{d1} \leq X_{c1}$

(22) $X_{d2} \leq X_{c1} + X_{c2}$

(23) $X_{d3} \leq X_{c1} + X_{c2} + X_{c3}$

(24) $X_{d4} \leq X_{c1} + X_{c2} + X_{c3} + X_{c4}$

(25) $X_{d5} \leq X_{c1} + X_{c2} + X_{c3} + X_{c4} + X_{c5}$

Por último, se tiene dos predecesores inmediatos, b y d, planteándose las restricciones así:

(26) $2X_{e1} \leq X_{b1} + X_{d1}$

(27) $2X_{e2} \leq X_{b1} + X_{d1} + X_{b2} + X_{d2}$

(28) $2X_{e3} \leq X_{b1} + X_{d1} + X_{b2} + X_{d2} + X_{b3} + X_{d3}$

(29) $2X_{e4} \leq X_{b1} + X_{d1} + X_{b2} + X_{d2} + X_{b3} + X_{d3} + X_{b4} + X_{d4}$

(30) $2X_{e4} \leq X_{b1} + X_{d1} + X_{b2} + X_{d2} + X_{b3} + X_{d3} + X_{b4} + X_{d4} + X_{b5} + X_{d5}$

Tabla 3. Resultado del ejemplo en WinQSB (Escobar Alvarán, Garcés Hincapié, & Restrepo Correa, 2012)

Decisión variable	Valor de la solución	Costo unitario o beneficio	Contribución total	Reducir costos
X_{a1}	1.00	1.00	1.00	0
X_{a2}	0	2.00	0	1.00
X_{a3}	0	3.00	0	2.00
X_{a4}	0	4.00	0	3.00
X_{a5}	0	5.00	0	4.00
X_{b1}	0	1.00	0	-3.50
X_{b2}	1.00	2.00	2.00	-1.00
X_{b3}	0	3.00	0	0
X_{b4}	0	4.00	0	1.00
X_{b5}	0	5.00	0	2.00
X_{c1}	1.00	1.00	1.00	-1.00
X_{c2}	0	2.00	0	0
X_{c3}	0	3.00	0	1.00
X_{c4}	0	4.00	0	2.00
X_{c5}	0	5.00	0	3.00
X_{d1}	0	1.00	0	-3.50
X_{d2}	0	2.00	0	-1.00
X_{d3}	1.00	3.00	3.00	0
X_{d4}	0	4.00	0	1.00
X_{d5}	0	5.00	0	2.00
X_{e1}	0	1.00	0	3.00
X_{e2}	0	2.00	0	-3.50
X_{e3}	0	3.00	0	-1.00
X_{e4}	1.00	4.00	4.00	0
X_e5	0	5.00	0	1.00
Valor	Objetivo	(Min.) =	11.00	

La Tabla 3 proporciona los resultados del ejercicio analizado (Escobar Alvarán, Garcés Hincapié, & Restrepo Correa, 2012). La columna uno representa las variables de decisión en su totalidad, la columna dos contiene el resultado mediante una representación binaria; cero si no se considera la variable y uno de lo contrario, por lo tanto, las variables contempladas en el resultado son X_{a1}, X_{b2}, X_{c1}, X_{d3}, X_{e4}. El resto de las variables son desestimadas en el resultado. Al sumar los costos de cada variable obtenemos como resultado 11.

Con base en los resultados la función objetivo arroja un resultado de 11, sin embargo, es irrelevante, ya que el fin real del problema es la minimización del número de máquinas o estaciones de trabajo (Escobar Alvarán, Garcés Hincapié, & Restrepo Correa, 2012).

Con el fin de conocer el uso de la programación lineal es notorio que, a pesar de ser un problema muy pequeño, resulta un número muy extenso de restricciones, esto mediante el software llamado “WinQSB”.

Debido a la complejidad computacional mencionada para los problemas como MAL y SALBP; utilizar métodos de búsqueda exhaustiva no es conveniente, por lo que en la siguiente sección se explora un heurístico ampliamente utilizado para resolver problemas de esta categoría.

2.7. Algoritmos Genéticos.

Los Algoritmos Genéticos son algoritmos de búsqueda heurística inspirados en la evolución natural de poblaciones, en la teoría de la selección natural de Darwin y en la teoría de la transferencia del material genético de Mendel.

Estos algoritmos son frecuentemente utilizados para la resolución de problemas complejos de optimización. La historia de los algoritmos genéticos se remonta a finales de la década de los sesenta, en la que John Holland y un grupo de investigadores de la Universidad de Michigan tenían el objetivo de estudiar formalmente el fenómeno de la adaptación como ocurre en la naturaleza y las posibles formas de aplicar este mecanismo de selección y adaptación natural en el campo de la computación.

Los algoritmos genéticos tal vez sean los más conocidos de todas las estrategias que se son parte del cómputo evolutivo. Éstos fueron propuestos por (Goldberg & Holland, 2022) en la década de los 60's. Los algoritmos genéticos son poblacionales y a diferencia de los algoritmos anteriormente mencionados, basan su búsqueda en la cruza (reproducción), i.e. consideran a todos los individuos parte de una misma especie y son capaces de generar hijos combinando su información. No por ello dejan de lado el proceso de mutación, pues cada que un hijo es generado es expuesto a dicho proceso, pero no siempre se cumple. Los algoritmos genéticos en un inicio son extintivos, esto es, la generación nueva reemplaza directamente a la generación anterior, sin embargo, una forma de elitismo puede agregarse, siempre manteniendo a la mejor solución dentro de la población actual. (Justo, 2019).

El libro de Holland "Adaptación en Sistemas Naturales y Artificiales", presentó por primera vez el Algoritmo Genético en 1975. Desde entonces muchos más se han dedicado a estudiar la forma en que estos algoritmos pueden aplicarse a diferentes problemas. La principal característica que hace atractiva la aplicación de los algoritmos genéticos son las pocas condiciones que debe cumplir el espacio de búsqueda; ya que, solo se requiere una forma de representación de las soluciones candidatas y una forma de medir el grado en el que cada solución candidatas resuelve el problema.

En abril del año 2012 por la universidad tecnológica de Pereira se presentó la Aplicación de la programación entera binaria para resolver el problema simple de balanceo de línea de ensamble: un caso de estudio desarrollado por Daniel Felipe Escobar Alvarán, Julián Alberto Garcés Hincapié, Jorge Hernán Restrepo Correa de Ingeniera Industrial en la Universidad Tecnológica de Pereira, Pereira, Colombia.

En el documento se presentó la forma reducida para la aplicación de la programación lineal para solucionar un problema simple de balanceo de líneas de ensamble. El problema de las líneas de ensamble ha sido estudiando muchas veces desde diferente enfoque utilizando algoritmos exactos y algoritmos heurísticos.

Dicho eso consistía en el objetivo el cual podría ser maximiza la eficiencia de la línea, minimizar el tiempo ocioso o minimizar el número de estaciones requeridas en la línea de ensamble, por lo tanto, este compuesto por una función objetivo y un conjunto de restricciones.

En una solución factible cada tarea se debe asignar exactamente a una estación, debe cumplir las relaciones de precedencia obligatoriamente y también no deben exceder el tiempo de ciclo asignado.

Por lo tanto, se divide en dos categorías; SALBPs los cuales son problemas simples, pocas variables de entrada por otra parte esta GALBPs donde son problemas generales donde existen casos más reales y complejos.

Debido a que el problema en esta tesis consiste en minimizar dos funciones objetivo: número de trabajadores y número de estaciones de trabajo, el problema es del tipo multi-objetivo. El método del óptimo Pareto es el un

enfoque que permite resolver este tipo de problemas. Los análisis de las Curvas de indiferencia y de la Caja de Edgeworth, propuestos por el economista, político y sociólogo italiano Vilfredo Pareto (1848-1923) quien desarrolló el concepto de Óptimo para aquella situación en la cual se cumple que no es posible beneficiar a una persona sin perjudicar a otra cuando se tienen problemas multi-objetivos. En la búsqueda de un equilibrio con mejor bienestar, ambos agentes aceptan el intercambio hasta el punto en que este deja de generar beneficios. El Óptimo de Pareto se basa en criterios de utilidad: si algo genera o produce provecho, comodidad, fruto o interés sin perjudicar a otro, despertará un proceso natural de optimización que permitirá alcanzar un punto óptimo.

Es preciso señalar que el óptimo de Pareto no es sensible a los desequilibrios e injusticias en la asignación de recursos, conocidos como dotación inicial, sean estos factores, bienes o servicios, dado que una situación en la que se distribuyan 10 unidades de un bien para su consumo entre dos individuos permite obtener 10 óptimos distintos de Pareto con independencia de la justicia de tal asignación. Son óptimos de Pareto tanto una distribución del tipo 10 a 0, como otra del tipo 5 a 5, ya que una vez asignados en ambos casos, para mejorar la situación de un individuo irremediablemente se empeoraría la situación del otro al tener que ceder una de las unidades del bien o servicio (aunque el primero parta de 0 y el último de 10).

Utilizando el análisis de las curvas de indiferencia, Vilfredo Pareto fue uno de los primeros economistas que buscó determinar científicamente dónde se encontraba el mayor bienestar alcanzable de una sociedad. La solución que encontró Pareto puede parecer simple, pero es de enorme profundidad: la máxima prosperidad común se obtiene cuando ninguna persona puede aumentar su bienestar en un intercambio sin perjudicar a otra. Basta, por lo tanto, que exista una posibilidad de intercambio en que dos personas ganen

para demostrar que ése no es el punto de máxima utilidad que se podría alcanzar.

El óptimo de Pareto es una herramienta de trabajo que se emplea en los procesos de negociación y en teoría de juegos porque ofrece dentro de sus límites, parámetros claros de decisión; pero estos mismos límites son los que le hacen recibir severos cuestionamientos por parte de economistas como Amartya Sen, y es que el Óptimo de Pareto no dice nada sobre la ética y la justicia. Por eso que la tarea de una función de bienestar social con la conjunción de ética y justicia distributiva está aún pendiente. (Moreno, 2011).

Capítulo 3. Metodología.

En esta investigación se realizó una búsqueda para la optimización de trabajadores y estaciones de trabajo en líneas de ensamble multi-tripuladas mediante algoritmos genéticos.

Se dio a conocer el método de solución en donde inicia con Descripción general del AG.

Se utilizaron algoritmos genéticos para resolver el MALB generando un número *n* de soluciones aleatorias. En donde 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 hace 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 que, 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.

3.1. Codificación y decodificación de solución

La manera de interpretar la información que está publicada en GitHub (https://github.com/juanseck/GAMmALB) se desarrolla como se muestra en la Figura 2.

Figura 2. Publicación en GitHub

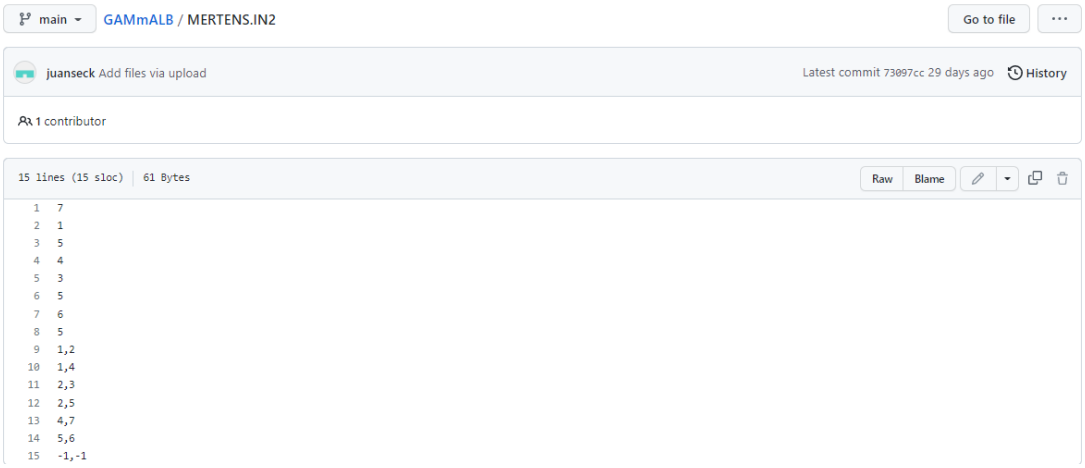
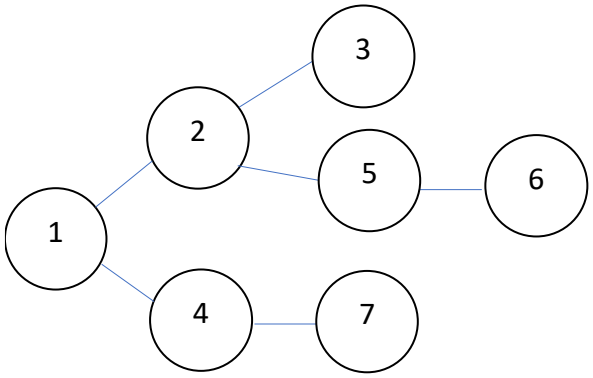


Tabla 4. Información de MERTENS para diagrama de precedencia.

Tareas	Tiempo	Predecesor
1	1	-
2	5	1
3	4	2
4	3	1
5	5	2
6	6	5
7	5	4

En la Tabla 4 se muestran los tiempos para cada tarea por procesar y la procedencia de esta.

Figura 3. Ejemplo de un diagrama de precedencia.



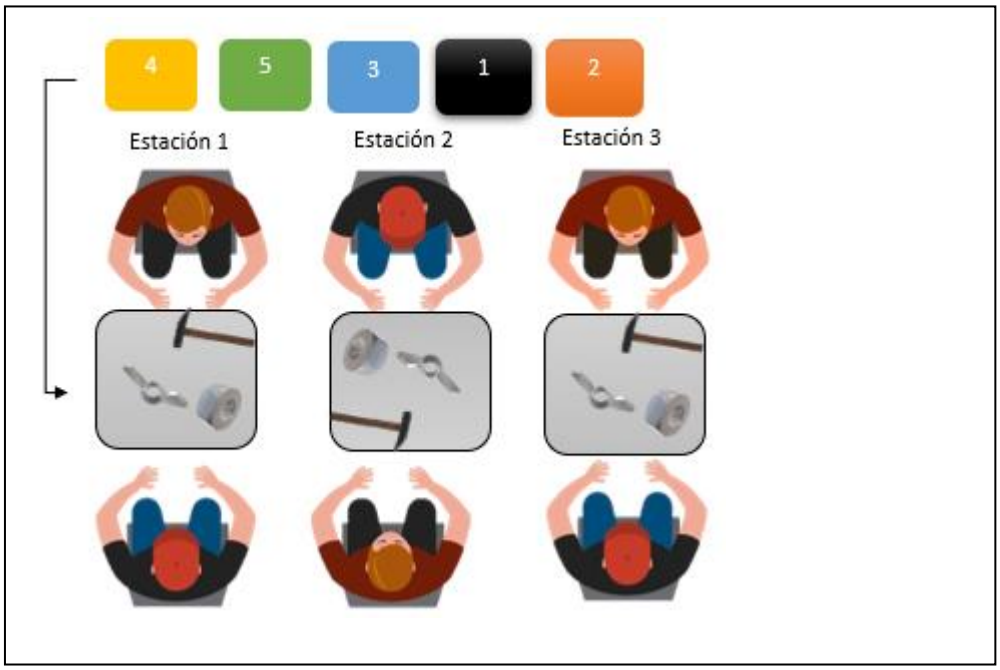
Fuente: Elaboración propia.

El diagrama de la Figura 3 muestra la forma de especificar los tiempos desde inicio hasta el fin, dando a conocer los tiempos necesario para las tareas, es decir, muestra las restricciones de precedencia: Las actividades de los nodos 2 y 4 no se pueden realizar si no ha concluido la actividad del nodo 1, además, la actividad 3 podrá realizarse hasta que concluya la actividad 2, actividad 5 es precedente de la 2 y la 6 precedente de la 5. Finalmente, la actividad 7 tiene como restricción de precedencia a la actividad 4. La totalidad de las restricciones pueden encontrarse en la Tabla 4.

En un inicio con 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 es el orden en que son procesados por el sistema. La Figura 4 muestra un ejemplo de cómo una

codificación de 5 trabajos se procesa en una línea multi-tripulada de 3 estaciones.

Figura 4. Ejemplo de codificación y decodificación de una secuencia de 5 trabajos.



Fuente: Elaboración propia.

Es posible que un trabajo no pueda realizarse, ya que se puede presentar el caso en que un trabajo no pueda realizarse ya que no se han 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

3.2 Función de costo y Algoritmo genético

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 en 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

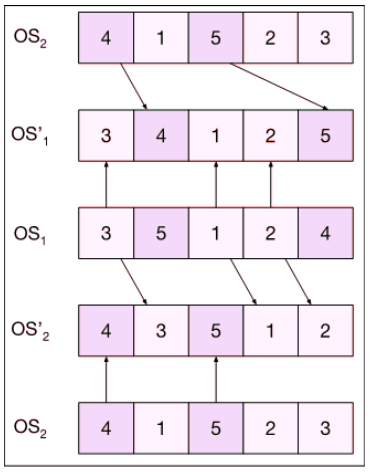
Para el apartado de selección, se utiliza primero elitismo, donde se toman las dos mejores soluciones (las que tenga un menor costo) como parte de la población refinada. Después, el resto de $n-2$ soluciones en la población se selecciona por medio de torneo. De la población original se eligen dos

soluciones de manera aleatoria, y se selecciona a 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 tal 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 vacias 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 llenado los espacios vacios de OS_2' con las operaciones de J_A en OS_1 en el orden en que aparecen. La Figura 5 presenta un ejemplo del cruce JBX para dos soluciones, cada una de 5 trabajos.

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

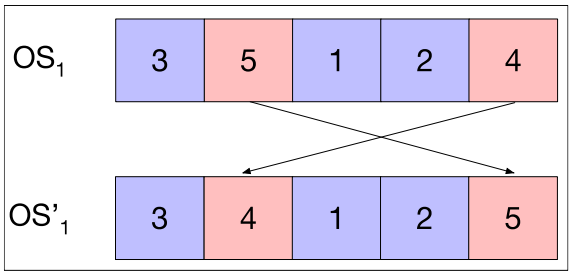


Fuente: Elaboración propia.

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 6 muestra un ejemplo de la mutación para una solución con 5 trabajos. Se observa la secuencia de trabajos OS_1, la cual es sometida a una mutación que tiene como resultado OS_1' ; las actividades 3, 1 y 2 se mantienen en su posición después de la mutación, intercambiando la posición de las tareas 5 y 4. Las posiciones de intercambio se definen de manera aleatoria.

Figura 6: Ejemplo de mutación por intercambio.

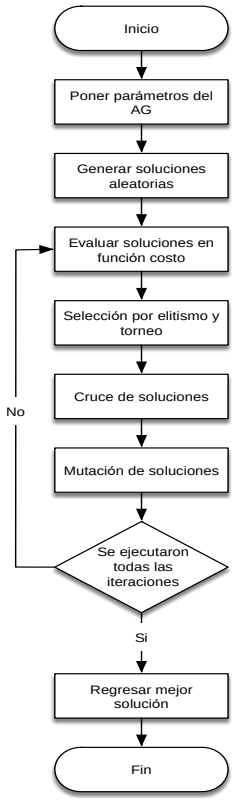


Fuente: Elaboración propia

De manera general el algoritmo propuesto funciona como se muestra en la Figura 7. Comienza reconociendo información de la instancia o problema correspondiente, posteriormente se definen los parámetros de los operadores genéticos tales como probabilidad de cruza, de mutación, tamaño de la población. El algoritmo genético comienza con la generación de soluciones aleatorias pero factibles. En este punto es necesario enfatizar que la forma en que se generan las soluciones factibles es una de las innovaciones en esta investigación ya que no es necesario depurarlas al ser soluciones viables. El siguiente paso consiste en evaluar las soluciones en función al costo. Dentro el operador genético denominado torneo se aplica el método de selección por

torneo y elitismo. Posteriormente se aplica el operador genético de cruza y finalmente el operador mutación. Estas actividades se repiten hasta el número de iteraciones predeterminadas, por lo que este es el método de paro del algoritmo.

Figura 7: Diagrama de flujo del AG



Fuente: Elaboración propia

Capítulo 4. Experimentación

4.1 Descripción de problemas de prueba

Para probar la efectividad del algoritmo propuesto, se realizó la toma de 11 problemas de prueba utilizados en la literatura especializada (Zamzam, Sadek, Afia, & El-Kharbotly , 2015)((Zamzam N. y., 2021) (Roshani & Giglio, 2020). Donde cada uno de estos problemas se prueban 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.

El problema de balanceo de líneas de ensamble multi-tripulado, tiene como objetivo minimizar el número de trabajadores y el número de estaciones de trabajo (Zamzam N. y., 2021). Se trata de un problema multi-objetivo, lo que implica la imposibilidad de encontrar un óptimo global para uno u otro objetivo en lo singular, por lo anterior el uso de algoritmos heurísticos como los AG son métodos apropiados que brindan buenas soluciones en tiempos computacionales factibles (Mares-Castro, 2021). Es importante señalar que el artículo de investigación de (Zamzam N. y., 2021) no describe el AG utilizado por lo que no se conoce el proceso específico de los operadores genéticos.

Otro método utilizado con la intención de optimizar los resultados del problema que sirve para comparar al algoritmo propuesto en esta tesis es la solución al Problema de balanceo de líneas de ensamble multi-tripuladas mediante el algoritmo de búsqueda Tabú. El cual considera a las plantas de producción que fabrican productos de gran tamaño y volumen, como automóviles y camiones, suelen encontrarse con problemas de equilibrio de líneas de montaje con personal múltiple (MALBP). En el artículo de (Roshani & Giglio, 2020), se tiene en cuenta una versión de MALBP orientada a los costos, a

saber, CMALBP. Este tipo de problemas pueden surgir en las líneas de montaje finales de productos en las que el proceso de fabricación requiere bastante mano de obra. Dado que CMALBP es NP-Hard, se desarrolla un enfoque heurístico basado en un algoritmo de búsqueda Tabú para resolver el problema. El algoritmo propuesto en el artículo próximo anterior utiliza dos mecanismos de generación de vecindad, a saber, intercambio y mutación, que colaboran eficazmente entre sí para proporcionar nuevas soluciones factibles.

Además, se utilizan dos listas tabú independientes (que corresponden a los dos mecanismos de generación mencionados) para comprobar si está prohibido o no pasar a una nueva solución vecina generada.

Para examinar la eficiencia del algoritmo propuesto en (Roshani & Giglio, 2020), se recopilaron y resolvieron algunos casos experimentales de la literatura. Los resultados obtenidos muestran la efectividad del enfoque de búsqueda tabú propuesto.

Por otra parte, un AG híbrido (Zamzam, Sadek, Afia, & El-Kharbotly , 2015) es comparado con el algoritmo de la presente tesis en el que se aborda el problema del equilibrio de una línea de montaje multi-personal (MAL). Esas líneas son importantes para el montaje de productos de mediano y gran tamaño como lavadoras, aires acondicionados, autobuses, camiones, helicópteros, entre otros. Además, se define un nuevo indicador (MPNW) para determinar el número máximo permitido de trabajadores en la estación. El número de estaciones que expresan una mejor utilización del área disponible se considera como criterio de rendimiento en este algoritmo. Para demostrar el rendimiento del algoritmo desarrollado, se prueba en instancias recopiladas de la literatura y muestra su eficacia para resolver este problema y proporciona resultados comparables.

Los parámetros utilizados en los operadores genéticos son: una población de tamaño 20, probabilidad de cruza de 0.8, la probabilidad de mutación es de 0.2. En referencia al operador genético cruza utiliza el propuesto por (Leu, Mathenson, & Rees, 1994).

Debido a que la literatura de comparación no cuenta con información precisa referente a la codificación de los algoritmos, éstos no se replican, únicamente se comparan con los resultados de la propuesta de esta tesis.

4.2 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 **Algoritmo Genético** se tomaron cuatro problemas con diferentes tiempos de ciclo del banco de prueba presentado en CITA.

4.3 Descripción de indicadores y pruebas estadísticas

Para realizar la comparación del 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 menor sean estos indicadores, mejor será la solución calculada.

Para el algoritmo propuesto se tomaron 30 corridas independientes por instancia, y se seleccionó a 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.

Tabla 5. Comparación de los resultados obtenidos por el algoritmo propuesto contra otros publicados para los problemas de prueba.

		Nessren, 2021		Roshani, 2013		Dimitriadis, 2006		Fattahi, 2011		Roshani, 2013		Zamzam, 2015		Algoritmo propuesto	
	CT	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS	NW	NS
Merten (7)	6	---	---	---	---	6	6	6	3	6	3	6	3	6	3
	7	---	---	---	---	5	5	5	3	5	3	5	3	5	3
	8	---	---	---	---	5	5	5	3	5	3	5	3	5	3
	10	---	---	---	---	3	3	4	3	3	3	3	3	3	2
	15	---	---	---	---	2	2	2	2	2	2	2	2	2	1
	18	---	---	---	---	2	2	2	1	2	1	2	1	2	1
Bowman(8)	17	---	---	---	---	---	---	5	5	5	5	5	5	5	3
	20	---	---	---	---	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	8	8	5	8	6	8	6	8	4
	7	---	---	---	---	7	7	7	5	7	6	7	6	7	4
	8	---	---	---	---	6	6	6	5	6	5	6	5	6	3
	10	---	---	---	---	4	4	4	4	4	4	4	4	4	2
	18	---	---	---	---	3	3	3	2	3	2	3	2	3	2
Jackson(11)	7	---	---	---	---	8	7	9	5	8	6	8	6	7	4
	9	---	---	---	---	6	5	6	4	6	4	6	4	6	3
	10	---	---	---	---	6	6	5	4	5	4	5	4	5	3
	13	---	---	---	---	4	4	4	3	4	3	4	3	4	2
	14	---	---	---	---	4	4	4	3	4	3	4	3	4	2
	21	---	---	---	---	3	3	3	2	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	---	---	---	---	9	9	8	7	8	7	8	7	8	4
	15	---	---	---	---	8	8	8	7	8	7	8	7	8	4
	21	---	---	---	---	5	5	5	5	5	5	5	5	5	3
	26	---	---	---	---	5	5	5	4	5	4	5	4	5	3
	35	---	---	---	---	3	3	3	3	3	3	3	3	3	2
	39	---	---	---	---	3	3	3	2	3	2	3	2	3	2
Heskia (28)	138	8	5	8	5	8	6	8	4	8	5	8	5	8	4
	205	5	4	5	4	6	6	5	3	5	3	5	3	6	3
	216	5	3	5	3	5	4	5	3	5	3	5	3	5	3
	256	4	3	5	3	5	5	4	3	4	3	4	3	4	2
	324	4	2	4	2	4	3	4	2	4	2	4	2	4	2
	342	3	2	3	2	3	3	3	2	3	2	3	2	3	2
Kilbridge(45)	57	10	6	10	6	10	8	10	5	10	6	10	6	10	5
	79	7	4	8	4	7	6	7	5	7	5	7	5	8	4

	92	6	4	7	4	6	5	7	4	6	4	6	4	6	3
	110	6	3	6	3	6	5	6	3	6	3	6	3	6	3
	138	4	3	4	3	4	4	4	3	4	3	4	3	4	2
	184	3	2	3	2	3	3	3	2	3	2	3	2	3	2
Tonge(70)	176	21	17	22	12	22	21	21	14	21	20	21	19	21	11
	364	10	6	10	6	10	9	10	5	10	7	10	7	10	5
	410	9	5	9	5	9	7	9	4	9	6	9	5	9	5
	468	8	4	8	5	8	7	8	4	8	4	8	4	8	4
	527	7	4	7	4	7	7	7	4	7	4	7	4	7	4
Arcus(83)	5048	16	11	16	12	16	16	16	11	16	11	16	11	16	8
	5853	14	10	14	10	14	13	14	10	14	10	14	9	14	7
	6842	13	8	12	8	12	10	12	8	12	8	12	8	12	6
	7571	11	9	11	10	11	11	11	7	11	7	11	7	11	6
	8412	10	8	10	8	10	10	10	6	10	6	10	6	10	5
	8998	10	5	9	7	9	8	9	6	9	6	9	6	9	5
	10816	8	6	8	5	8	8	8	6	8	5	8	5	8	4
Arcus (111)	5755	---	---	---	---	27	24	27	14	27	20	27	21	27	14
	8847	18	12	18	14	18	18	18	12	18	12	18	12	18	9

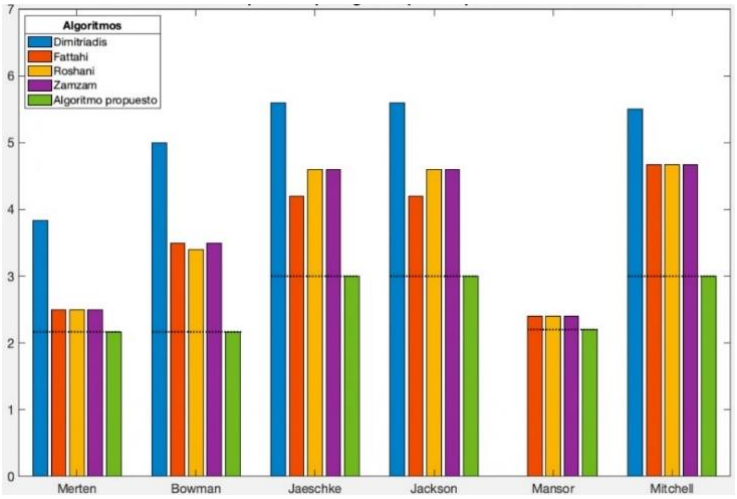
	10027	16	10	16	12	16	15	16	10	16	10	16	11	16	8
	10743	15	14	15	14	15	14	15	10	15	10	15	10	15	8
	11378	14	8	14	9	14	9	14	7	14	9	14	9	14	7
	17067	9	5	9	7	9	7	9	5	9	6	9	6	9	5

4.4. Análisis y Resultados

En la Tabla 5 se observa que en la mayoría de las instancias el algoritmo propuesto obtiene 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 (en color azul) y mejoró los resultados en 36 instancias marcadas en color rojo dentro de la Tabla 5, lo que muestra la eficacia del algoritmo para balancear la línea de producción de manera adecuada en este tipo de problemas.

La Figura 8 nos permite identificar el valor esperado de las estaciones de trabajo para 6 instancias, ésta nos muestra que en todas las instancias el promedio de este indicador es menor que los promedios presentados por Dimitriadis, (2006); 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.

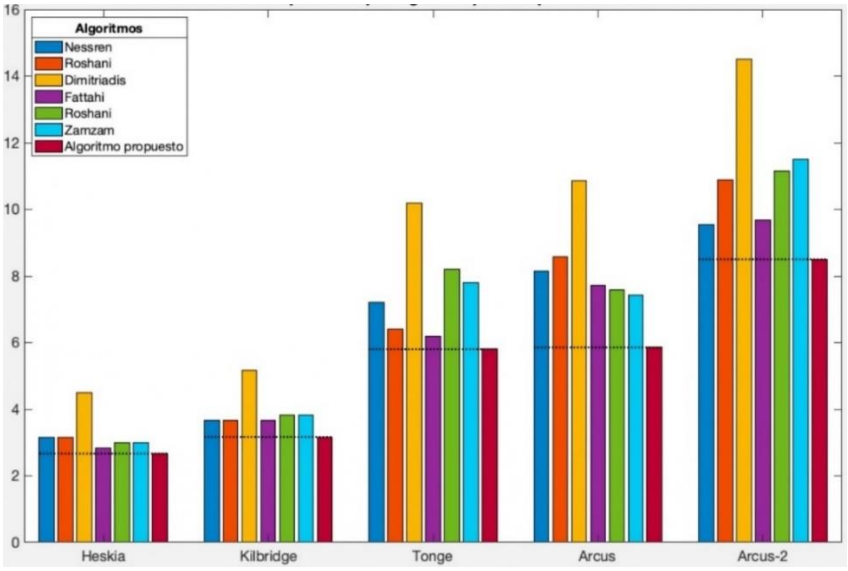
Figura 8. Número de estaciones promedio por algoritmo para los problemas Merten a Mitchell



Fuente: Elaboración propia

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 8 en todos los casos, el valor esperado de estaciones de trabajo es menor para el algoritmo propuesto en este documento.

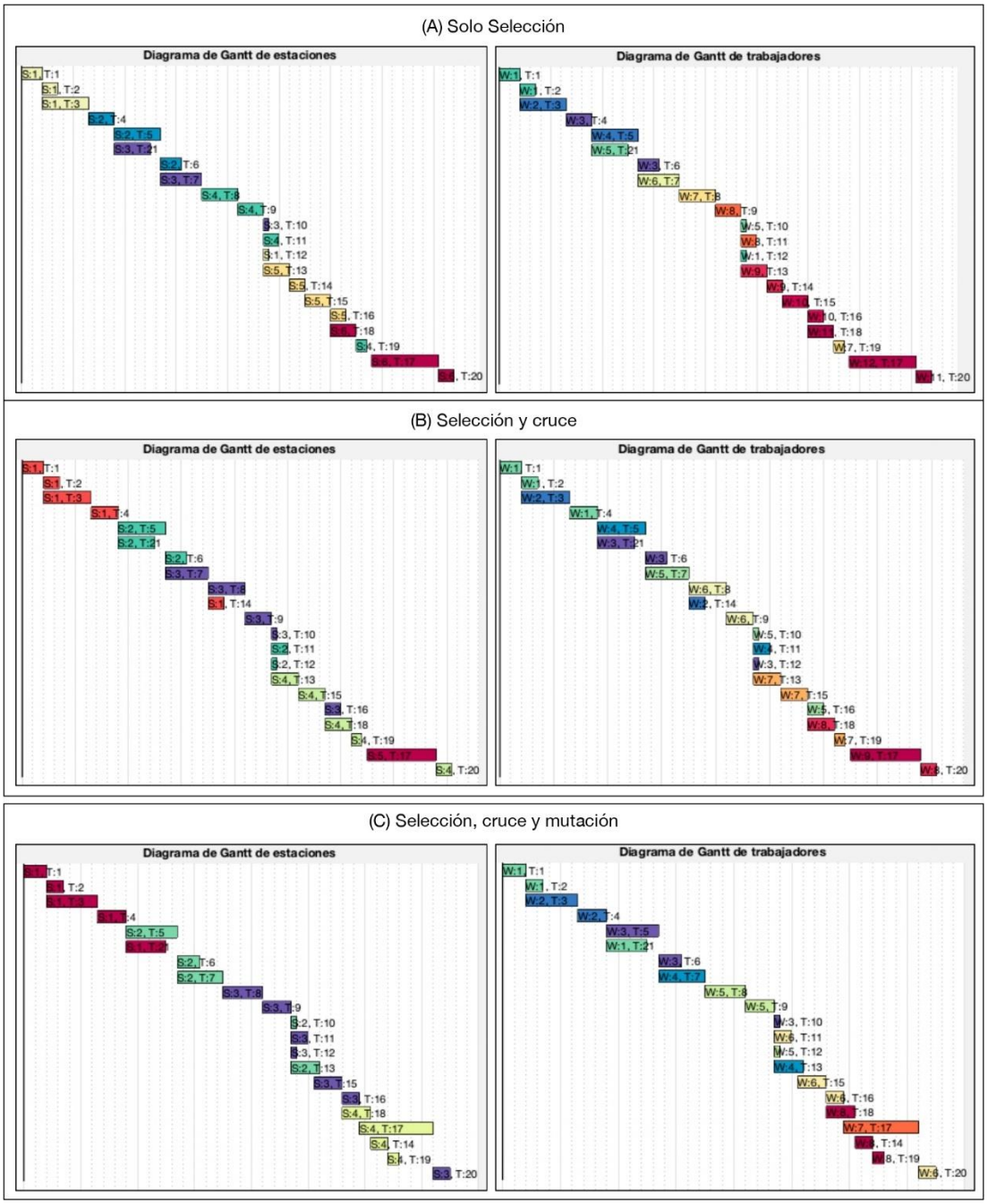
Figura 9. Número de estaciones promedio por algoritmo para los problemas Heskia a Arcus-2



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 10, la cual contiene los diagramas de Gantt para la instancia Mitchell con un tiempo de ciclo de 14 unidades. La Figura 10.A muestra los resultados de la asignación de tareas a los trabajadores para el operador genético de selección. La Figura 10.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 al que previamente se procesó los operadores selección y cruza, como se puede observar el la Figura 10.C. Este último operador genético tiene la intención de evitar óptimos locales.

Figura 10. Diagramas de Gantt de los trabajadores

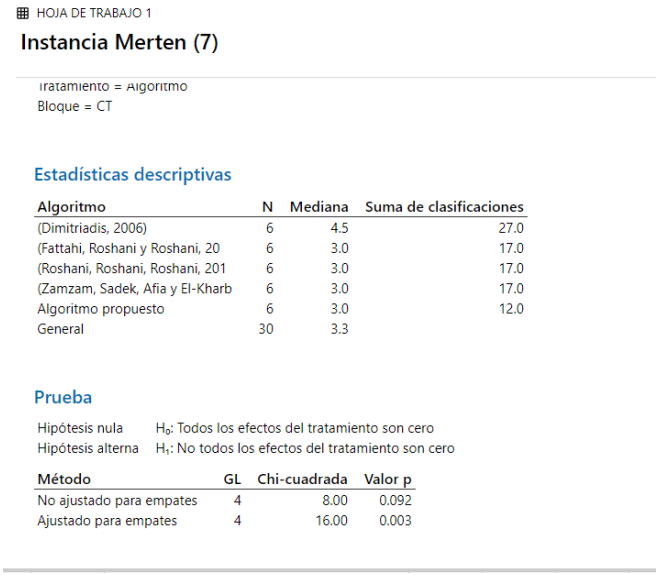


Fuente: Elaboración propia.

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 Optimiación 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 estadísticas 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 que la hipótesis nula no se acepta, es decir, que las medianas no son iguales como se puede observar en la Figura 11. Adicionalmente se observa que la mediana del algoritmo propuesto es menor que la de (Dimitriadis, 2006).

Figura 11. 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, 20	6	3.0	17.0
(Roshani, Roshani, Roshani, 201	6	3.0	17.0
(Zamzam, Sadek, Afia y El-Kharb	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 éste generó los mejores resultados.

Figura 12. 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, 20	6	4.00	18.0
(Roshani, Roshani, Roshani, 201	6	4.00	18.0
(Zamzam, Sadek, Afia y El-Kharb	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 13. 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 13. 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, 20	5	4.8	13.0
(Roshani, Roshani, Roshani, 201	5	5.0	16.0
(Zamzam, Sadek, Afia y El-Kharb	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 14. Ésta muestra como resultado de la prueba de hipótesis un valor P de 0.000 no aceptando a la hipótesis nula, por lo que 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 14. 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, 20	6	3.5	16.5
(Roshani, Roshani, Roshani, 201	6	3.5	18.0
(Zamzam, Sadek, Afia y El-Kharb	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 15 indican que no existe una diferencia significativa entre los métodos utilizados.

Figura 15. 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, 20	5	2	13.0
(Roshani, Roshani, Roshani, 201	5	2	13.0
(Zamzam, Sadek, Afia y El-Kharb	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

La instancia Mitchell (21) fue comparada con los algoritmos mostrados en la Figura 16; la hipótesis nula no se acepta en el nivel de significancia de 0.05, por lo que las medianas son diferentes, siendo el algoritmo propuesto la menor con 3.1.

Figura 16. 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, 20	6	4.5	18.5
(Roshani, Roshani, Roshani, 201	6	4.5	18.5
(Zamzam, Sadek, Afia y El-Kharb	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

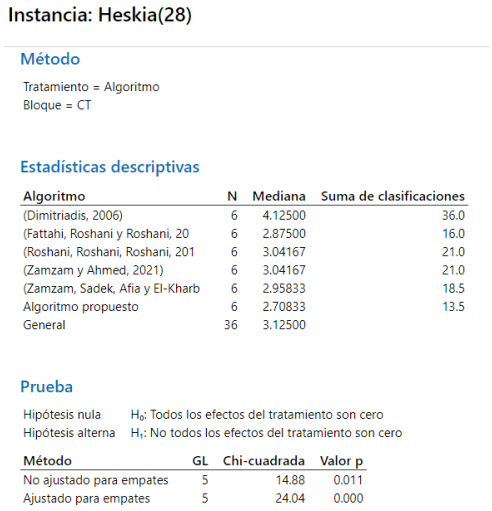
Método	GL	Chi-cuadrada	Valor p
No ajustado para empates	4	12.80	0.012
Ajustado para empates	4	19.69	0.001

Fuente: Elaboración propia utilizando el software Minitab 21.3.0

En la Figura 15 se muestras 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 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 17. 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, 20	6	2.87500	16.0
(Roshani, Roshani, Roshani, 201	6	3.04167	21.0
(Zamzam y Ahmed, 2021)	6	3.04167	21.0
(Zamzam, Sadek, Afia y El-Kharb	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 18. 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, 20	6	3.52083	19.0
(Roshani, Roshani, Roshani, 201	6	3.43750	19.0
(Zamzam y Ahmed, 2021)	6	3.43750	19.0
(Zamzam, Sadek, Afia y El-Kharb	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 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 que el de Dimitriadis, (2006); sin embargo, no existe diferencia significativa que el resto de los algoritmos de la literatura mostrados en la Figura 17.

Figura 19. 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, 20	5	4.16667	10.5
(Roshani, Roshani, Roshani, 201	5	5.00000	18.5
(Zamzam y Ahmed, 2021)	5	4.83333	15.5
(Zamzam, Sadek, Afia y El-Kharb	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 al 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.

Figura 20. 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, 20	7	7.7500	24.0
(Roshani, Roshani, Roshani, 201	7	8.2500	28.0
(Zamzam y Ahmed, 2021)	7	8.2500	25.5
(Zamzam, Sadek, Afia y El-Kharb	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 21. 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, 20	5	9.9167	11.5
(Roshani, Roshani, Roshani, 201	5	12.2500	25.5
(Zamzam y Ahmed, 2021)	5	10.2500	15.5
(Zamzam, Sadek, Afia y El-Kharb	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.

5. Conclusiones.

Con la presencia de este nuevo algoritmo genético para la optimización del número de trabajadores y de estaciones de trabajo en una línea de ensamble multi-tripulada. El AG 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 AG para la generación de mejores soluciones.

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.

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 multiobjetivo, 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

Andrea. (19 de 07 de 2019). *educadictos*. Obtenido de https://www.educadictos.com/aparicion-de-la-produccion-en-serie/

Aspiazu, C. (21 de 05 de 2010). *Elitismo en algoritmos genéticos*. Obtenido de mente errabunda: http://menteerrabunda.blogspot.com/2010/05/elitismo-en-algoritmos-geneticos.html#:~:text=El%20mecanismo%20elitista%20pretende%20asegurar,intacta%2C%20sin%20recombinarse%20ni%20mutarse.

Browman. (June de 1960). Assambly line balancing by linear programming. *Oper*, págs. 385-389.

Cristofani, F. (2023). *Líneas de ensamble.* Obtenido de deingenieriaindustrial.com: https://deingenieriaindustrial.com/administracion-operaciones/lineas-de-ensamble/

Díaz, A. G. (1996). *Optimización Heurística y Redes Neuronales.* Madrid .

Feliciano, R. R. (02 de 2020). *Matemática Serie 23*. Obtenido de https://matte23.blogspot.com/2020/02/las-mutaciones.html

Goldberg, D. E., & Holland, J. (2022). Genetic Algorithms and Machine Learning. *Machine Learning*, 95-99.

González-Jaime, A.-K. A.-I. (2017). *ecorfan.* Obtenido de https://www.ecorfan.org/republicofperu/research_journals/Revista_de_Ingeniería_Industrial/vol1num2/ECORFAN_Revista_de_Ingenier%C3%ADa_Industrial_V1_N2_1.pdf

Groover, M. (2007). Fundamentos de Manufactura Moderna . En M. Groover, *Fundamentos de Manufactura Moderna* (pág. 1380).

Ibrahim H, O., & James P, K. (1996). *Meta-Heuristics: Theory and Applications.* Springer.

Justo, A. E. (2019). Un algoritmo inspirado en la naturaleza para resolver problemas de optimización numérica restringida con alta dimensionalidad. *Centro de Investigación en Inteligencia Artificial*.

Kumar, N. y. (2013). Assembly Line Balancing: A Review of Developments and Trends in Approach to Industrial Application. *Global Journal of Researches in Engineering*, 28-50.

Leu, Y.-y., Mathenson, L., & Rees, L. (1994). Assembly Line Balancing Using Genetic Algorithms with Heuristic-Generated Initial Populations and Multiple Evaluation Criteria. *Decision Sciences*, 581-606.

López, J. F. (15 de 04 de 2020). *economipedia*. Obtenido de https://economipedia.com/definiciones/taylorismo.html#:~:text=El%20aporte%20fundamental%20de%20Taylor,la%20autonom%C3%ADa%20temporal%20del%20trabajador.&text=As%C3%AD%20pues%2C%20en%20primer%20lugar,tareas%20para%20aumentar%20la%20productividad.

Mares-Castro, A. (2021). Robust parameter design and economical multi-objective . *optimization on characterizing rubber for shoe soles*, 160-169.

Moreno, M. A. (09 de 02 de 2011). *elblogsalmon*. Obtenido de https://www.elblogsalmon.com/conceptos-de-economia/que-es-un-optimo-de-pareto

Paredes-Quevedo, J., Alpala, L. O., Soto-Chávez, L. E., & León-Batallas, A. (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 Tecnica De La Facultad De Ingenieria Universidad Del Zulia*, 48-57.

Pena-Orozco, D. L.-G. (2019). *Problema de balanceo de una línea del tipo SALBP: caso de una línea de confección de prendas.* Obtenido de https://doi.org/10.22335/rlct.v11i2.866.

Petterson, J. H. (February de 1975). Assembly Line Balancing Zero-One programming with Fibonacci Search. *Oper*, págs. 166-172.

Raquel Murillo Garcia, F. P.-L. (06 de 2018). *eumed*. Obtenido de https://www.eumed.net/rev/oel/2018/06/ensamble-balanceo-productividad.html

Riquett Rodríguez, A. H. (2021). *Corporación Universidad de la Costa.* Obtenido de https://repositorio.cuc.edu.co/bitstream/handle/11323/8457/Propuesta%20de%20mejora%20para%20incrementar%20la%20productividad%20en%20la%20fabricaci%c3%b3n%20de%20placas%20expandidas%20de%20bater%c3%adas%20plomo-%c3%a1cido%20a%20trav%c3%a9s%20de%20balanceo%

Roshani, A. y. (2012). Multi-manned Assembly Line Balancing Problem: Minimizing Cycle Time. *Proceedings of the IIE Asian Conference 2012*, 612-620.

Roshani, A., & Giglio, D. (2020). A Tabu Search Algorithm for the Cost-Oriented Multi-Manned Assembly Line Balancing Problem. *International Journal of Industrial Engineering & Production Research*, 189-202.

Sato-Michels, A. T.-L. (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.

Scholl A, B. N. (2008). The sequence-dependent assembly line balancing problem. *O Spectrum*, 579–609.

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

Tello, D. E. (31 de 01 de 2018). *Algoritmos de búsqueda exhaustiva.* Obtenido de CINVESTAV-Tamaulipas: https://www.tamps.cinvestav.mx/~ertello/algorithms/sesion07.pdf

Thangavelu, S. a. (March de 1971). Assembly Line Balancing by Xero-One Integer Programming. *AIIE Trans*, págs. 61-68.

Torres-Medina , Y. (16 de 07 de 2020). *redalyc.* Obtenido de https://www.redalyc.org/journal/5537/553768213006/

W.W, W. (april de 1961). Commrnts on s Prprt by Bowman. *Ooper*, págs. 274-276.

Yazdanparast, V. y. (2011). «Multi-manned production Lines with labor Concentration. *Australian Journal of Basic and Applied Sciences*, 839-846.

Zamzam, N. y. (2021). Time and Space Multi-Manned Assembly Line Balancing Problem Using Genetic Algorithm. *Journal of Industrial Engineering and Management*, 733-749.

Zamzam, N. y. (2021). Time and Space Multi-Manned Assembly Line Balancing Problem Using Genetic Algorithm. *Journal of Industrial Engineering and Management*, 733-749.

Zamzam, N. y.-M. (2021). 733-749.

Zamzam, N., Sadek, Y., Afia, N., & El-Kharbotly , A. (2015). Multi-Manned Assembly Line Balancing using Genetic Algorithm. *International Journal of Engineering Research & Technology*, 56-61.