DECANATURA DE INGENIERÍA INDUSTRIAL
MAESTRÍA EN INGENIERÍA INDUSTRIAL
FORMATO DE ENTREGA TRABAJO DE GRADO

Fecha de entrega:

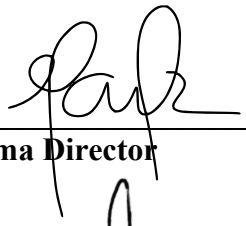
Estudiante: Ángela Patricia Guerra Cruz

Director: Jose Fernando Jiménez

Codirector: Sonia Alexandra Jaimes

El presente documento avala la entrega del trabajo de grado por parte del director y codirector.

Documentos anexos: copia digital del Trabajo de Grado (1).



Firma Director

Firma Codirector

Firma Estudiante

OPTIMIZACIÓN BI-OBJETIVO DE EFICIENCIA EN TIEMPO DE PRODUCCIÓN Y CONSUMO DE ENERGÍA EN ENTORNOS FLEXIBLES DE MANUFACTURA JOB-SHOP

Ángela Patricia Guerra Cruz

Escuela Colombiana de Ingeniería Julio Garavito
Decanatura de Ingeniería Industrial
Maestría en Ingeniería Industrial
Bogotá D.C., Colombia
2023

OPTIMIZACIÓN BI-OBJETIVO DE EFICIENCIA EN TIEMPO DE PRODUCCIÓN Y CONSUMO DE ENERGÍA EN ENTORNOS FLEXIBLES DE MANUFACTURA JOB-SHOP

Ángela Patricia Guerra Cruz

Trabajo de grado presentado como requisito para optar al título de
Magister en Ingeniería Industrial

Director
Ing. Jose Fernando Jimenez Gordillo, PhD.

Codirector
Ing. Sonia Alexandra Jaimes Suarez, M. Sc.

Escuela Colombiana de Ingeniería Julio Garavito
Decanatura de Ingeniería Industrial
Maestría en Ingeniería Industrial
Bogotá D.C., Colombia
2023

Resumen

Con este trabajo se presenta un nuevo enfoque para abordar objetivos de minimizar el tiempo de ejecución de operaciones y el consumo de energía en entornos de manufactura flexibles. El método propuesto se basa en un Algoritmo Genético de Ordenación No Dominado II (NSGA-II), el cual permite encontrar eficientemente soluciones óptimas entre estos dos objetivos críticos. La clave del método es una técnica de ordenación no dominado que clasifica e identifica eficazmente las soluciones Pareto óptimas. El rendimiento del algoritmo se evaluó experimentalmente en casos de referencia conocidos, como Kacem y Brandimarte. Los resultados demuestran la eficacia del enfoque al momento de minimizar simultáneamente el tiempo de ejecución y el consumo de energía. Además, la diversidad inherente a las soluciones de Pareto contribuye a una optimización equilibrada de estos objetivos. Los resultados muestran el potencial de los algoritmos genéticos, y en particular del NSGA-II, como herramientas convenientes para abordar retos de optimización multiobjetivo. Este estudio acerca la brecha entre los conocimientos teóricos y la aplicación práctica y demuestra la importancia de incluir más objetivos en escenarios reales, haciendo énfasis en las prometedoras perspectivas de estas técnicas en escenarios complejos reales.

Abstract

This paper presents a novel approach to address the dual objectives of minimizing makespan and energy consumption in flexible manufacturing processes. The proposed method leverages the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) to efficiently find optimal trade-offs between these two critical objectives. The cornerstone of the approach is a non-dominated sorting technique, which effectively ranks and identifies optimal Pareto solutions. The algorithm's performance was evaluated through experimentation on well-known benchmark instances, including Kacem and Brandimarte. The results demonstrate the efficacy of the approach in simultaneously minimizing both makespan and energy consumption. Furthermore, the inherent diversity within the Pareto solutions contributes to a balanced optimization of these objectives. The findings underscore the potential of genetic algorithms, particularly NSGA-II, as proficient tools to tackle multiobjective optimization challenges. This study bridges the gap between theoretical insights and practical implementation and demonstrates the importance of including more objectives in real scenarios, emphasizing the promising prospects of such techniques in complex real-world scenarios.

Tabla de contenido

Lista de tablas

Lista de Figuras

1 Introducción

1.1 Problemática de estudio y justificación

En la actualidad y debido a la Cuarta Revolución Industrial, estamos viviendo una transformación, sin precedentes, de la tecnología digital y física. Esta transformación, ha traído consigo cambios significativos en las industrias, ya que, para lograr un impacto positivo y tener reconocimiento en el mercado, ha sido necesario acogerse a las facetas de la, hoy conocida, Industria 4.0.

El término Industria 4.0, se refiere a un nuevo modelo de organización y control de la cadena de valor, que se apoya y es posible gracias a las tecnologías de la información y a la transformación digital (Del Val Román, 2018). A nivel organizacional, los cambios conceptuales se perciben desde los productos y servicios, que en este nuevo modelo se basan en la diferenciación, hasta la organización y cultura para la cual la digitalización empieza a significar un reto. Para las empresas, la transición hacia este nuevo modelo representa complejidad, ya que, requiere cambios que implican idear nuevas formas de crear valor en las organizaciones, por medio de la diferenciación, la tecnología, las operaciones inteligentes, la digitalización, entre otros (López Ramón y Cajal & Escudero Ceballos, 2016). Sin embargo, cabe mencionar que esta transición no es opcional, si el objetivo de una empresa es mantenerse en el mercado y ser competitiva.

Satisfacer las necesidades del mercado actual, implica basarse en la personalización y la creación de productos y servicios innovadores. Los clientes exigen calidad, pero están dispuestos a pagar, más por la experiencia o el servicio que por el producto o bien que adquieren. Mientras los gustos y preferencias del mercado sean altamente cambiantes, las empresas e industrias deben dirigir sus sistemas de producción hacia la personalización masiva. Este concepto se refiere a la capacidad de proporcionar productos y servicios diseñados individualmente a cada cliente, por medio de procesos integrados, ágiles y altamente flexibles (Da Silveira et al., 2001). En busca de mantenerse en el mercado y ser competitivas, las empresas e industrias deben lograr que sus productos y servicios estén actualizados, sean innovadores y ofrezcan consigo experiencias individualizadas. La Industria 4.0 está logrando generar valor en las operaciones y crear ambientes y estrategias, en los que, como parte de la eficiencia operacional, se tengan en conceptos como la flexibilidad y la sostenibilidad (López Ramón y Cajal & Escudero Ceballos, 2016), para lograr, además del óptimo desarrollo de las actividades, la satisfacción y fidelidad de los clientes.

Estos retos afectan directamente la planeación y programación de operaciones, por ello, los problemas de programación y secuenciación han tomado importancia para las industrias. Estos permiten responder eficientemente a requerimientos de los clientes, tales como tiempos de espera más cortos y entregas más rápidas (C. J. Liao et al., 2013), facilitando, además, la toma de decisiones operativas. Teniendo en cuenta lo anterior, el presente trabajo de investigación está enfocado en un

Flexible Job Shop Problem (FJSP), un problema clásico de programación y secuenciación de operaciones.

El FJSP tiene un papel fundamental en la programación y secuenciación de operaciones, debido a que el ambiente de manufactura que lo caracteriza es altamente flexible. Hasta ahora, la flexibilidad en estos entornos es una característica que ha sido abarcada en la investigación, sin embargo, el desarrollo continuo de la tecnología, aunque ha mejorado la calidad de vida en muchos aspectos, ha causado también grandes daños en el medio ambiente (Z. Jiang et al., 2014). Daños que han ocasionado que hoy en día, la sostenibilidad ambieexintal sea vista como un factor influyente en la toma de decisiones. Los constantes requerimientos de la sociedad han hecho que el consumo de energía se convierta en un problema importante para el mundo, llegando a ser uno de los mayores desafíos que debe enfrentar la industria manufacturera. La industria es un consumidor intensivo de energía (T. Jiang & Deng, 2018) según la Administración de Información de Energía de EE. UU. (EIA), en el 2015 dicha industria era responsable de la mitad del consumo total de energía del mundo, consumo que casi se había duplicado en los últimos 60 años y estaba creciendo a una tasa del 1,2% anual, tasa que también iba en aumento (Wu & Sun, 2018).

En el caso de Colombia, para el 2015, el sector industrial manufacturero tenía un consumo cercano al 30% de la energía final del país, convirtiéndose en el segundo sector más consumidor después del transporte, como mostrado en la Figura 1.

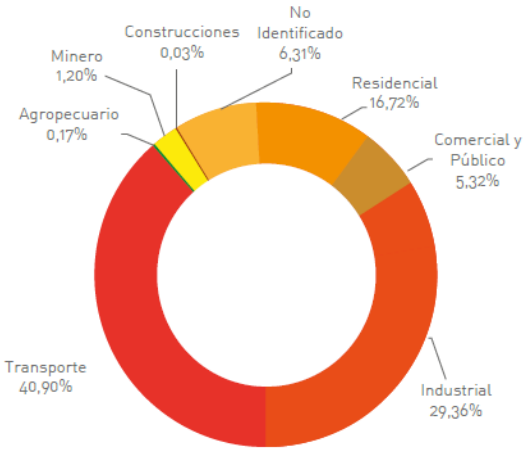


Figura 1. Participación de los usos de la energía eléctrica en la industria (República de Colombia et al., 2016).

Del 29,36% de consumo energético final en la industria manufacturera anteriormente mostrado, el 17% estaba caracterizado como energía eléctrica, y como puede apreciarse en la Figura 2, el 75,9% del uso de la energía eléctrica estaba constituido como fuerza motriz (República de Colombia et al., 2016).

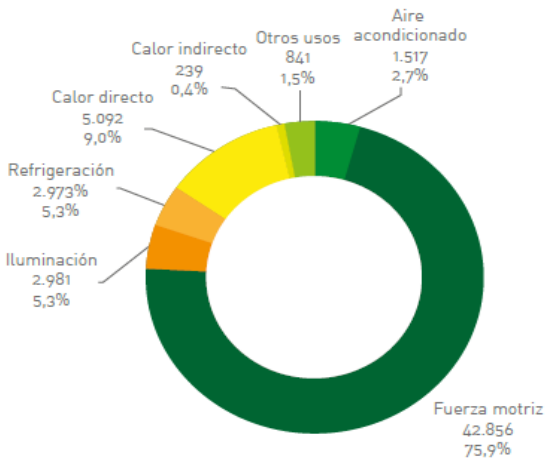


Figura 2. Distribución del consumo energético final (República de Colombia et al., 2016).

Independiente del consumo energético que demanda, la industria manufacturera juega un papel central en el desarrollo de la economía. Según cifras del Departamento Administrativo Nacional de Estadística (DANE), en el 2018 este sector tuvo una participación del 12,8% del total del Producto Interno Bruto (PIB) nacional. Este porcentaje posicionaba al sector como el tercero de mayor relevancia económica en Colombia. Teniendo en cuenta la información precedente, es claro que no es viable económicamente detener o mitigar el crecimiento industrial, pero sí es primordial, explorar mecanismos de eficiencia energética para reducir el consumo de energía, ahorrar costos y apoyar actividades de fabricación amigables con el medio ambiente (Dai et al., 2019).

En consecuencia, uno de los mayores desafíos para la industria manufacturera, es lograr una adecuada planeación y optimización de sus operaciones, considerando la relación opuesta entre la eficiencia energética y otros objetivos de producción enfocados en el beneficio económico. Este desafío se hace aún mayor en un entorno FJSP, ya que las características que presenta, principalmente la alta flexibilidad en las máquinas, lo convierten en uno de los entornos industriales con mayor consumo de energía (Mokhtari & Hasani, 2017).

A medida que crece la conciencia ambiental, la programación y secuenciación de actividades sostenible está atrayendo cada vez más atención. Se ha demostrado que tener en cuenta el ahorro de energía desde la programación, ayuda a reducir el consumo de esta, sin necesidad de que las empresas obtengan nuevos equipos o técnicas, ni rediseñen sus estructuras (Meng et al., 2019). Teniendo en cuenta lo anterior, es importante que las industrias manufactureras realicen la gestión de la planeación de producción, considerando y buscando un balance entre la efectividad y la eficiencia de las operaciones. La efectividad haciendo referencia al *makespan* o tiempo de finalización total y la eficiencia relacionada con la disminución del consumo energético. Por lo tanto, este trabajo se enfoca en lograr un balance idóneo entre los objetivos de efectividad y eficiencia en un FJSP.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un modelo de programación de producción de un ambiente de manufactura *Flexible Job Shop*, en el que se logre la minimización y un balance simultaneo entre el *makespan* y el consumo energético de la producción.

1.2.2 Objetivos específicos

- Realizar una recopilación y análisis de la investigación en el campo de la programación de producción con inclusión de características de consumo energético.
- Representar matemáticamente el modelo de programación y secuenciación del problema *Flexible Job Shop* bi-objetivo.
- Proponer un modelo de programación basado en un algoritmo genético de ordenamiento no dominado (NSGA-II) como método de solución del problema *Flexible Job Shop* bi-objetivo.
- Evaluar el desempeño del modelo propuesto en relación con las métricas asociadas al tiempo de terminación de operaciones (*makespan*) y al consumo energético.

1.3 Alcances y limitaciones del trabajo de investigación

En este apartado se especifican los compromisos que se encuentran dentro y fuera del alcance del presente trabajo de investigación.

1.3.1 Compromisos dentro del alcance

- Programación de una metaheurística basada en un algoritmo genético de ordenamiento no dominado (NSGA-II), como solución para el problema *Flexible Job Shop* considerando dos objetivos, eficiencia medida como *makespan* y efectividad, considerada como consumo energético.
- Evaluación el desempeño del modelo de solución propuesto a partir de comparaciones halladas en la literatura, que permitan evaluar la calidad de las respuestas obtenidas y los tiempos computacionales.

1.3.2 Compromisos por fuera del alcance

- No se presentará integración de la solución propuesta con un sistema de manufactura real.

1.4 Metodología

En el desarrollo de este trabajo de grado se implementó una metodología secuencial que permitió abordar cada uno de los objetivos propuestos. A continuación, en la Figura 3, se detalla cada uno de los pasos:

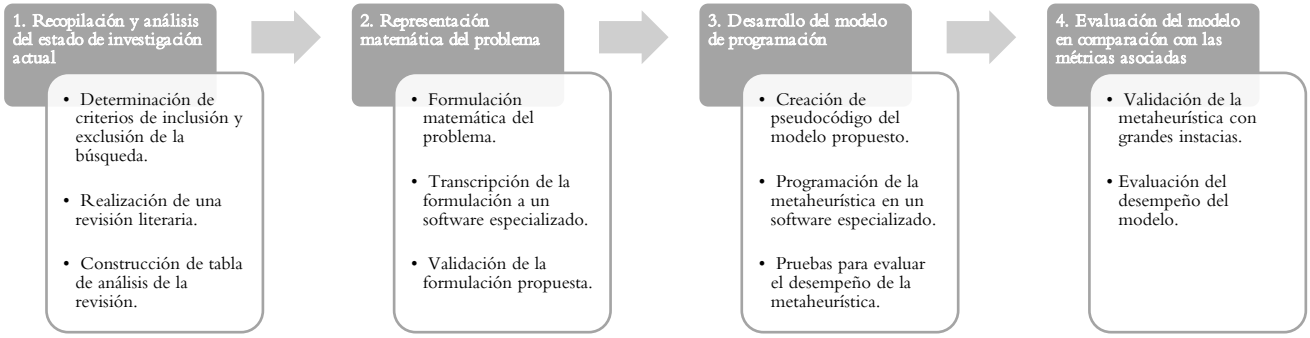


Figura 3. Metodología propuesta para el desarrollo del trabajo.

2 Revisión literaria del problema Flexible Job Shop con inclusión de características ambientales

En este capítulo se presenta la revisión de la literatura relacionada con programación de la producción que incluye enfoques ambientales. El objetivo principal es caracterizar los trabajos realizados en el campo, de forma tal que sea posible identificar las particularidades de cada investigación analizada, en términos de objetivos trazados, entornos de manufactura implicados, características incluidas y métodos de solución propuestos.

Para que los resultados de este proceso de revisión y análisis cumplieran con las expectativas y objetivos planteados, se define una metodología de búsqueda y clasificación de artículos que se detallará en la primera sección de este capítulo. En la segunda parte, se presenta el concepto formal del problema de programación de producción. En tercer lugar, una breve introducción de los métodos de solución existentes para el problema de programación de producción y, por último, se exponen las características que han llevado a que la solución del problema en estudio se enfoque en argumentos ambientales y se realiza una caracterización y categorización de los artículos analizados y los métodos de solución propuestos en cada uno de estos artículos.

2.1 Proceso definido para la revisión literaria

El proceso bajo el cual se realiza la revisión literaria, parte de definir los criterios de inclusión y exclusión de la búsqueda, para ello, se definen las palabras claves bajo las cuales se ejecutará la búsqueda y otros aspectos de inclusión como bases de datos y años de publicación de los artículos.

Teniendo en cuenta lo anterior, se realiza una revisión de las publicaciones de las bases de datos Google Scholar, IEEE Xplore, Science Direct, Scopus y Springer Link. los criterios de búsqueda se basan en combinaciones de las palabras clave definidas: *job shop problem, flexible job shop, multi-objective scheduling, multi-objective optimization, energy consumption, energy efficient, green manufacturing, genetic algorithm, NSGA-II.* Se seleccionan artículos publicados entre 2013 y 2022.

De la búsqueda resultan 67 artículos. La depuración se basa en un análisis del enfoque de cada artículo, se decide no incluir artículos que abordaran problemas mono-objetivo o aquellos que, aunque abordaban un problema multiobjetivo, entre sus objetivos no se encuentra ninguno de carácter ambiental. Producto de esta depuración, resultan 26 artículos a los que se les realiza un análisis y se identifican: la estructuración de la problemática, la formulación matemática y el método de solución propuesto en cada uno de ellos.

2.2 Problema de programación de producción

El *Job Shop Problem* (JSP), es un problema de programación de producción clásico en investigación de operaciones, que es catalogado como un problema complejo de optimización combinatoria y *NP-hard* en términos de complejidad computacional (Garey & Johnson, 1979).

En el JSP se cuenta con un conjunto finito de n trabajos $J = \{J_1, J_2, J_3, \ldots, J_n\}$ que son procesados por un conjunto finito de m máquinas $M = \{M_1, M_2, M_3, \ldots, M_m\}$. Cada trabajo J_i, consiste en una secuencia de n_j operaciones $O_{i1}, O_{i2}, \ldots, O_{in}$. La operación O_{ij} debe ser procesada sin interrupciones en la máquina $M_{ij} \in M$, durante un tiempo de procesamiento TP_{ij}. Las operaciones $O_{i1}, O_{i2}, \ldots, O_{in}$, deben ser procesadas una después de otra en el orden dado y cada máquina puede procesar una única operación a la vez (Chan et al., 2006). El problema radica entonces, en determinar el orden de procesamiento de los trabajos en cada una de las máquinas, de forma tal, que se optimice el valor objetivo (Mastrolilli & Gambardella, 2000).

El *Flexible Job Shop Problem* (FJSP) es una generalización del JSP, que proporciona una aproximación más cercana a una amplia gama de problemas encontrados en sistemas de fabricación reales. Existen dos tipos de FJSP, el *Flexible Job Shop Total* (TFSJP), en donde cada operación puede ser procesada en cualquier máquina disponible, y el *Flexible Job Shop Parcial* (PFJSP), en el que existen restricciones en cuanto a las máquinas en las que se pueden procesar las operaciones (Marzouki et al., 2017), es decir, una operación no puede procesarse en todas las máquinas.

Existen $(n_1)!$ $(n_2)!\ldots$ $(n_m)!$ posibles soluciones teóricas, aunque no todas son factibles, siendo n_k el número de operaciones a ser realizadas en la máquina k, por lo que el FJSSP se considera también *NP-hard*. La mejor solución debe satisfacer dos condiciones: respetar las restricciones de precedencia del trabajo y optimizar la medida de referencia (Sule, 2021). Por lo que este problema puede ser analizado desde dos subproblemas:

1) Subproblema de enrutamiento: consiste en asignar cada operación a una máquina de un conjunto de máquinas posibles.
2) Subproblema de programación: consiste en secuenciar las operaciones asignadas a una máquina para lograr una programación viable que optimice los objetivos definidos.

Teniendo en cuenta lo anterior, el problema radica en determinar tanto una secuenciación, como una asignación de las operaciones en todas las máquinas, de forma tal que se optimicen los objetivos definidos y se respeten las restricciones planteadas (Saad et al., 2007). Teóricamente, el FJSP está formado bajo los siguientes planteamientos (Xia & Wu, 2005):

1. Un número k de máquinas denotadas como $M = \{M_1, M_2, \ldots, M_k\}$.
2. Un conjunto de i trabajos independientes $J = \{ J_1, J_2, J_3, \ldots, J_i\}$.

3. Un trabajo Ji está conformado por una secuencia n de operaciones denotadas como (Oi1, Oi2, …, Oij).
4. Cada trabajo i está compuesto por una secuencia de N_j operaciones O_{ij}, i = 1,2, …, n y j = 1,2, …, N_j.
5. Para cada operación O_{ij} hay un grupo de máquinas capaces de realizarla denotado por $M_{ij} \subseteq M$.
6. El tiempo de procesamiento de cada operación es dependiente de la máquina. Se denota TP_{ijk} para el tiempo de procesamiento de la operación O_{ij} realizada en la máquina M_k.

Se consideran, además, los siguientes supuestos (Chaudhry & Khan, 2016):

a. Un trabajo está terminado cuando todas las operaciones de este fueron ejecutadas.
b. Asignar una operación O_{ij} a una máquina M_k ocasiona que esta última esté ocupada durante todo el tiempo de procesamiento TP_{ijk}.
c. Todas las máquinas y todos los trabajos están disponibles en el tiempo t = 0.
d. Cada máquina puede ejecutar solo una operación al tiempo.
e. No hay restricciones de precedencia entre las operaciones de los diferentes trabajos.
f. Una vez una operación es iniciada no puede ser interrumpida.
g. El tiempo de transporte de un trabajo entre máquinas y el tiempo de alistamiento de la máquina para el procesamiento de una operación, están incluidos en el tiempo de procesamiento.

2.3 Métodos de solución para el problema de programación de producción

Se han propuesto varias maneras de resolver el problema de programación de producción FJSP. En esta sección se presentan los métodos y tipos de solución existentes, haciendo énfasis en los métodos metaheurísticos y en el algoritmo genético de ordenamiento no dominado (NSGA-II), metaheurística que se utilizará como solución al problema planteado en esta investigación.

Los métodos de solución para problemas de programación de producción están divididos, como se puede apreciar en la Figura 4, en métodos de optimización exacta y métodos de aproximación. Los métodos de optimización exacta garantizan encontrar la mejor solución posible y están divididos en métodos constructivos, compuestos por reglas exactas y métodos enumerativos de los que hacen parte la programación matemática y algoritmos de ramificación y poda (B&B). Por su parte, aunque los métodos de aproximación no garantizan la mejor solución, han tomado fuerza en la investigación porque proveen buenas soluciones en tiempos computacionales razonables, característica importante si se tiene en cuenta la complejidad de problemas como el FJSP (Amjad et al., 2018). Como parte de los métodos de aproximación, se encuentran las heurísticas constructivas y mejoradas, los métodos de búsqueda local y los algoritmos metaheurísticos.

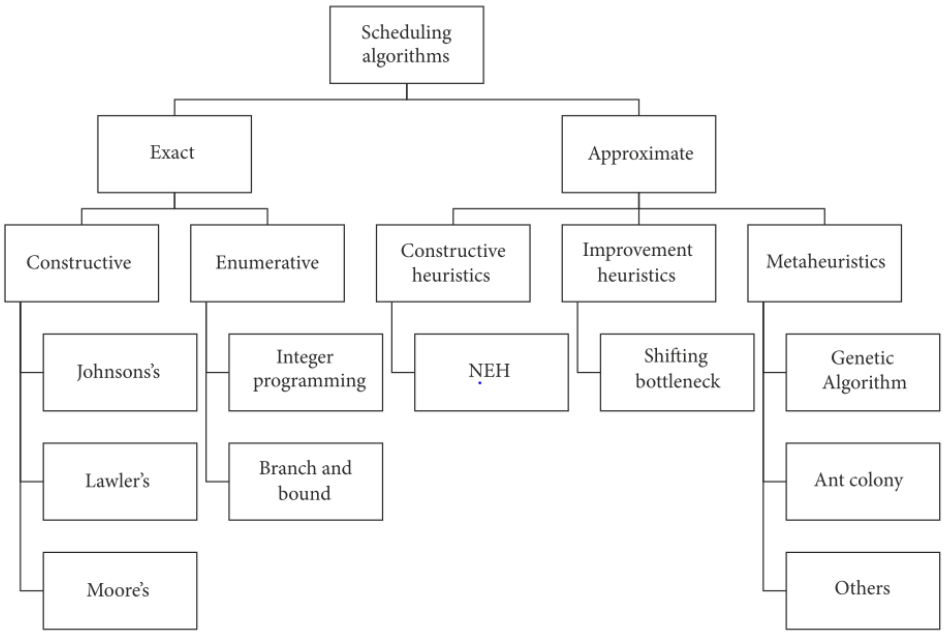


Figura 4. Algoritmos de solución para problemas de programación de producción (Amjad et al., 2018).

2.3.1 Métodos exactos

Desde 1950, la programación matemática y la investigación de operaciones se han aplicado para obtener soluciones óptimas globales o determinísticas. Los primeros enfoques en este campo fueron los métodos constructivos basados en reglas exactas, en estos se establecen una serie de reglas que determinan el orden de procesamiento de las entradas y es posible obtener una solución óptima exacta y posteriormente los métodos enumerativos, basados en programación matemática (J. Zhang et al., 2019).

Johnson (1954), propuso una serie de reglas para resolver el problema flow shop de dos máquinas, a estas reglas se les denomina las reglas de Johnson. Wagner (1959), encontró algunas técnicas de programación matemática que podían resolver el problema de programación de forma óptima, sin embargo, tomaban un tiempo computacional considerable. En 1960, (Manne, 1960) tomo un segundo enfoque para los métodos exactos y mezcló programación lineal y entera discreta (MIP). Propuso entonces, una formulación matemática con una función objetivo lineal, una serie de restricciones también lineales y variables enteras binarias, que tomaba menos tiempo computacional que la de Wagner, sin embargo, también tenía sus limitaciones en cuanto a número de variables y trabajos a procesar.

En términos generales, los métodos exactos ya sean reglas definidas o programación matemática, pueden lograr la solución óptima en tiempos computacionales razonables, de problemas de programación de producción específicos. Solo los problemas de pequeña escala, con poca complejidad en cuanto a restricciones u objetivos, pueden resolverse por métodos exactos.

2.3.2 Métodos de aproximación

Con el constante desarrollo de las ciencias computacionales y los algoritmos inteligentes, los métodos de investigación y solución de problemas de programación de producción evolucionaron a los métodos de aproximación.

Heurísticas constructivas y mejoradas

Con estos métodos se puede hallar de forma rápida una solución al problema de programación de producción. Típicamente se incluyen tres métodos: reglas de prioridad de envíos, algoritmos de inserción y heurísticas basadas en cuello de botella.

Metaheurísticas

Los primeros algoritmos metaheurísticos utilizados para la solución de problemas de programación de producción, son el algoritmos genético (Genetic Algorithm - GA) y el método de búsqueda Tabú (Jones et al., 2002), sin embargo, existen otros algoritmos como el algoritmo de colonia de hormigas (Ant Colony Optimization - ACO), el algoritmo de enjambre de partículas (Particle Swarm Algorithm - PSA), el algoritmo de evolución diferencial (Differential Evolution Algorithm - DEA), el algoritmo de luciérnaga (Firefly Algorithm - FA), entre otros, que también se aplican en la solución de este problema.

Para solucionar el problema de programación de producción planteado en esta investigación, existen también algoritmos metaheurísticos, entre los que destacan los Algoritmos Evolutivos Multiobjetivo (MOEA). Los MOEA son una nueva generación de algoritmos metaheurísticos desarrollados para solucionar problemas de optimización multiobjetivo. Se clasifican en tres tipos:

Funciones agregadas

Método en el cual las funciones objetivo se combinan en un solo objetivo, es ideal con la opinión del tomador de decisiones. Un ejemplo de esto es el Algoritmo evolutivo multiobjetivo modificado y basado en descomposición, que propusieron (E. Jiang & Wang, 2019) como solución al FJSP multiobjetivo en el que se buscaba optimizar el *makespan* y el consumo energético.

Enfoques basados en Pareto

Categoría en la que se incorpora la optimización de Pareto en la formulación. Estos algoritmos no necesitan la opinión del tomador de decisiones, sin embargo, en su salida proponen una variedad de soluciones no dominadas o no comparables, que pueden ser utilizadas por diferentes tomadores de decisión. (Rahmati et al., 2013) proponen dos algoritmos de esta clase para solucionar el problema MOFJSP. El primero, un NSGA-II y el segundo, un NRGA. Algoritmos que son dos de los más conocidos en esta clase.

Enfoques basados en la población

En estos métodos, la población del algoritmo evolutivo se utiliza para diversificar la búsqueda, dividiendo la población en diferentes subpoblaciones que permitan la búsqueda en diferentes direcciones.

Algoritmo genético de ordenamiento no dominado NSGA-II

El algoritmo genético de ordenamiento no dominado (NSGA) es uno de los primeros algoritmos evolutivos capaz de encontrar, en una única corrida, múltiples soluciones no dominadas ubicadas en el frente óptimo de Pareto. El algoritmo, fue propuesto por Deb (1999). Posteriormente, el NSGA-II (Deb et al., 2002) surgió como mejora del NSGA. Los bajos requisitos computacionales de este algoritmo y la eficiencia que es capaz de alcanzar, gracias a una buena distribución de soluciones y a una mejor convergencia cerca del óptimo de Pareto, son algunas de las características por las cuales el NSGA-II se ha convertido en uno de los más conocidos MOEA's.

La base del algoritmo NSGA-II es el algoritmo genético (GA), en el cual, por medio de una función objetivo, se seleccionan, cruzan, mutan individuos que se comparan con los hijos o padres surgidos del proceso anterior, con el fin de permitir que sobrevivan aquellos que presenten una mejor función objetivo (Luke, 2013). El NSGA-II se diferencia del GA, ya que la selección de los individuos en el NSGA-II se hace basada en la no dominancia de las soluciones obtenidas al evaluar cada individuo en las funciones objetivo planteadas (Deb et al., 2002).

El NSGA-II está basado en la idea de transformar M objetivos en un único objetivo (*fitness*). Sin embargo, dadas las características multiobjetivo del problema, se genera un conjunto de soluciones denominado conjunto de soluciones óptimas de Pareto. Una solución pertenece a dicho conjunto si no es posible mejorar uno de los objetivos sin generar un deterioro en el otro objetivo evaluado (Talbi, 2009). Una vez se obtiene el conjunto de soluciones de Pareto, se dice que dicho conjunto forma una frontera de Pareto como solución para el problema de optimización propuesto. Las soluciones que se encuentran en la frontera son aquellas que no son dominadas por ninguna otra solución existente como se muestra en la Figura 5.

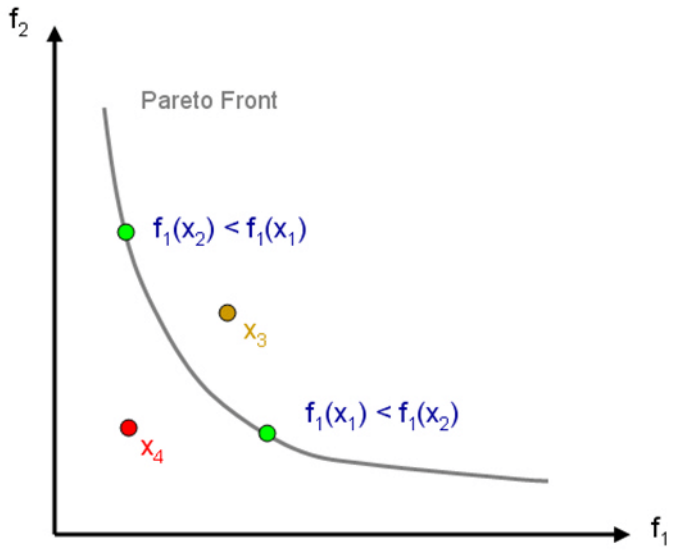


Figura 5. Frontera de Pareto en problemas de programación multiobjetivo (Kaoutar & Mohamed, 2017).

La obtención de la medida de desempeño (*fitness*) se logra por medio de la creación de un número de fronteras, que son seleccionadas según su no dominación. Las fronteras son producto de la generación de cromosomas que son evaluados según operadores de selección y elitismo determinados. Se asigna el fitness correspondiente a cada frontera y el proceso se repite hasta que todas las soluciones tengan una asignación y luego se determinan las distancias de aglomeración.

La Figura 6 muestra el proceso de generación de soluciones (operador del algoritmo genético), la realización de los puntos extremos de cada frontera y las soluciones que hacen parte de esta, y el corte que se realiza de acuerdo con la selección de mejorares soluciones (*survival the fittest*).

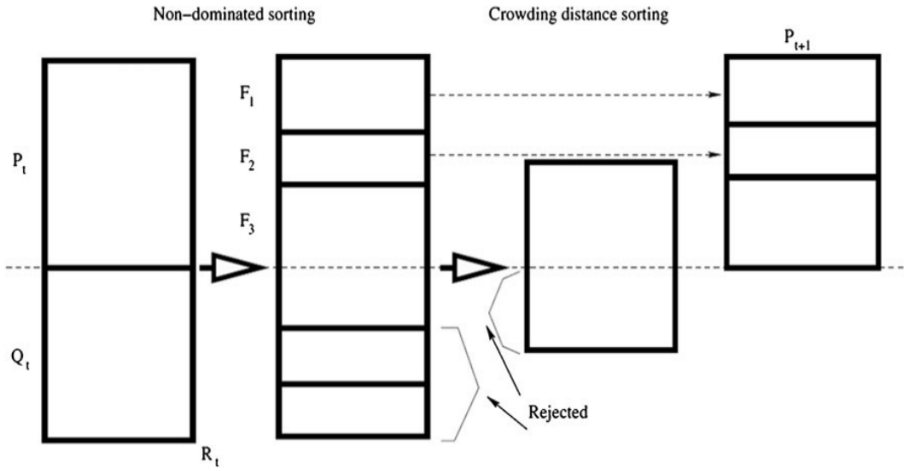


Figura 6. Proceso de elitismo del Algoritmo NSGA-II (Rahmati et al., 2013).

Con respecto al uso del NSGA-II como método de solución para el FJSP con consideraciones ambientales, (Wu & Sun, 2018)sugirió un problema en el que se consideran

velocidades de procesamiento variables, decisiones de encendido/apagado y tiempos ociosos como factores que consumen energía y añadió como parte de los objetivos, la optimización del encendido/apagado de las máquinas. (Z. Jiang et al., 2014) añadió al problema ambiental un factor de calidad de procesamiento que relacionó directamente con el factor costo. Por último, (Seng et al., 2018) destacó en su investigación el tema ambiental desde la medición de la huella de carbono, a lo que también sumó, velocidades de procesamiento variables en operaciones.

2.4 Revisión literaria: Flexible Job Shop multiobjetivo con enfoque ambiental

La sociedad ha evolucionado, en gran parte, gracias al continuo desarrollo de la productividad, sin embargo, esto ha ocasionado enormes daños al medio ambiente y deteriorado la sostenibilidad de nuestro planeta. Lo anterior, ha ocasionado que sea primordial diseñar estrategias que permitan crecer en productividad a la par que se minimiza la destrucción a la naturaleza (Z. Jiang et al., 2014).

Una forma adecuada de reducir el consumo de energía en los sistemas de fabricación es desarrollar estrategias de optimización, conscientes en disminuir los impactos ocasionados por los recursos utilizados para la producción (Li et al., 2020) esto teniendo en cuenta la relación opuesta entre la eficiencia energética y otros objetivos de producción enfocados en el beneficio económico. El consumo de energía se torna en un enfoque aún más crítico en el entorno de producción que representa el FJSP, debido a sus características, principalmente la alta flexibilidad en las máquinas, que lo convierten en uno de los entornos industriales con mayor consumo energético.

En la Tabla 2 se presenta una caracterización de los artículos que hicieron parte de esta revisión literaria. En la tabla se resumen características o particularidades propias de cada investigación, los objetivos y métodos de solución planteados en cada caso. A continuación, se describen los elementos principales.

a) Particularidades relacionadas con el consumo de energía. Estas particularidades relacionan la cantidad de energía utilizada y el rendimiento o resultado deseado. Adicionalmente, las emisiones de gases de efecto invernadero y la contaminación del aire son algunos de los indicadores medidos, ya que, se asocian con un alto consumo de energía no sostenible.
b) Método de solución. Algoritmo o enfoque utilizado para resolver la situación propuesta y alcanzar el objetivo específico. En el contexto del consumo de energía y la eficiencia del sistema, el método de solución generalmente se enfoca en minimización de consumo y maximización de eficiencia.
c) Función objetivo relacionada con la eficiencia del sistema. La literatura evaluada presenta las medidas clásicas de eficiencias en sistemas *Flexible Job Shop*. Algunas de ellas se presentan en la Tabla 1.

Tabla 1. Medidas de desempeño utilizadas en los problemas de programación.

Medida	Notación	Formula	Definición	Impacto
Makespan	C_{max}	$\max_{1 \le j \le n} C_j$	Tiempo necesario para completar todos los trabajos	Minimiza directamente los costos.
Tardanza total	T	$\sum_{j=1}^{n} T_j$	La diferencia positiva entre el tiempo de finalización de un trabajo y el tiempo de entrega.	A usar cuando trabajos retrasados son penalizados.
Tardanza media	$\bar{T}$	$\frac{\sum_{j=1}^{n} T_J}{n}$	El promedio de la tardanza de todos los trabajos.	A usar cuando se estipula un tiempo de entrega para la producción.
Carga total de las máquinas	W_T	$\sum_{j=1}^{n} W_j$	Carga total de trabajo de todas las máquinas.	Asegura una máxima utilización de máquinas.
Consumo de energía	EC_T	$\sum_{j=1}^{n} W_j e_j$	Energía total consumida por todas las máquinas.	Permite realizar la producción minimizando el consumo energético.

Tabla 2. Caracterización de la literatura.

	(a)										(b)									(c)					
(a) Particularidades relacionadas con el consumo de energía. (b) Método de solución. (c) Función objetivo relacionada con la eficiencia del sistema.	Energía de elementos independientes	Energía consumida en mantenimiento	Emisiones de carbono	Calidad relacionada con el costo de procesamiento	Inclusión de tiempo de alistamiento	Consumo total de energía	On/Off	Mantenimiento	Máquinas en modo stand-by	Algoritmo evolucionario multiobjetivo [illegible]	Algoritmo genético multiobjetivo (MOGA)	Algoritmo de competencia Imperialista (ICA)	Búsqueda de vecindad variable (VNS)	Algoritmo genético de ordenamiento no dominado (NSGA- II)	Optimización de enjambre de partículas (PSO)	Algoritmo de salto de rana barajado (SFLA)	Otros	Makespan	Consumo de energía	Tardanza (Tardiness)	Carga total de trabajo	Huella de carbono	Otros
(Zhou et al., 2022)			X														X	X	X						
(Liu et al., 2021)			X	X						X							X	X		X					
(Liang et al., 2021)						X								X							X	X			
(Xu et al., 2021)			X											X			X	X	X						
(J. Wang et al., 2020)				X													X		X			X			
(Lei et al., 2019)						X						X	X					X		X					
(Hemmati Far et al., 2019)						X								X						X					
(E. Jiang & Wang, 2019)										X								X	X						
(T. Jiang & Deng, 2018)	X																X		X	X					
(Wu & Sun, 2018)					X	X	X		X	X				X				X	X					X	

	(a)										(b)									(c)					
(a) Particularidades relacionadas con el consumo de energía. (b) Método de solución. (c) Función objetivo relacionada con la eficiencia del sistema.	Energía de elementos independientes	Energía consumida en mantenimiento	Emisiones de carbono	Calidad relacionada con el costo de procesamiento	Inclusión de tiempo de alistamiento	Consumo total de energía	On/Off	Mantenimiento	Máquinas en modo stand-by	Algoritmo evolucionario multiobjetivo [illegible]	Algoritmo genético multiobjetivo (MOGA)	Algoritmo de competencia Imperialista (ICA)	Búsqueda de vecindad variable (VNS)	Algoritmo genético de ordenamiento no dominado (NSGA- II)	Optimización de enjambre de partículas (PSO)	Algoritmo de salto de rana barajado (SFLA)	Otros	Makespan	Consumo de energía	Tardanza (Tardiness)	Carga total de trabajo	Huella de carbono	Otros
(Guo & Lei, 2018b)						X							X					X		X				
(Guo & Lei, 2018a)						X						X						X		X				
(Piroozfard et al., 2018)											X									X		X		
(W. Liao & Wang, 2018)		X			X			X	X		X							X				X		
(Seng et al., 2018)										X				X				X	X			X		
(X. Zhang et al., 2018)							X									X		X	X					
(H. Wang et al., 2018)											X				X			X	X					
(Mokhtari & Hasani, 2017)		X			X	X		X			X						X	X	X				X	
(L. Zhang et al., 2017)							X						X					X	X					
(Salido et al., 2017)					X						X							X	X					X
(Yin et al., 2017)					X						X							X	X					X
(Lei et al., 2017)									X							X			X		X			

	(a)										(b)									(c)					
(a) Particularidades relacionadas con el consumo de energía. (b) Método de solución. (c) Función objetivo relacionada con la eficiencia del sistema.	Energía de elementos independientes	Energía consumida en mantenimiento	Emisiones de carbono	Calidad relacionada con el costo de procesamiento	Inclusión de tiempo de alistamiento	Consumo total de energía	On/Off	Mantenimiento	Máquinas en modo stand-by	Algoritmo evolucionario multiobjetivo	Algoritmo genético multiobjetivo (MOGA)	Algoritmo de competencia Imperialista (ICA)	Búsqueda de vecindad variable (VNS)	Algoritmo genético de ordenamiento no dominado (NSGA- II)	Optimización de enjambre de partículas (PSO)	Algoritmo de salto de rana barajado (SFLA)	Otros	Makespan	Consumo de energía	Tardanza (Tardiness)	Carga total de trabajo	Huella de carbono	Otros
(R. Zhang & Chiong, 2016)											X								X	X				
(Singh et al., 2016)															X			X		X				X
(Z. Jiang et al., 2014)				X						X				X				X	X					X
(Rahmati et al., 2013)										X				X			X	X				X		X

La mayoría de los autores enfocan su investigación en resolver el problema planteándose principalmente dos objetivos, el *makespan* y el consumo energético y para esto se han propuesto distintos métodos de solución. (E. Jiang & Wang, 2019) sugieren un Algoritmo Evolucionario Multiobjetivo (MOEA) basado en descomposición e intensificación local. (Liang et al., 2021) presentan dos modelos de programación lineal entera mixta, en ellos, tuvieron en cuenta tiempo y energía ociosos. Además, diseñan un algoritmo de Búsqueda de Vecindad Variable (VNS) con dos estrategias, posponer y apagar/encender. (Xu et al., 2021)plantean un Algoritmo de Salto de Rana Barajado (SFLA) con una estrategia encendido/apagado para el ahorro de energía. En su modelo tienen en cuenta el consumo energético por procesamiento, tiempos ociosos, transporte y encendido/apagado de las máquinas. (H. Wang et al., 2018) propusieron un método hibrido en el que integran un Algoritmo Genético Multiobjetivo (MOGA) y la Optimización de Enjambre de Partículas (PSO) en un entorno dinámico.

Se presenta también el caso en el que el *makespan* y la tardanza, ambas medidas de eficiencia, se han tomado como objetivo de las investigaciones. A causa de esto, el argumento ambiental se ha añadido en las restricciones del problema. (Guo & Lei, 2018a) y (Lei et al., 2019) agregan como restricción un umbral de consumo de energía máximo. Los primeros desarrollaron dos investigaciones independientes, una en la que resuelven el problema mediante una Búsqueda de Vecindad Variable (VNS) de dos fases y otra en la que desarrollan un Algoritmo de Competencia Imperialista (ICA) en el que, como primera medida, añaden el consumo de energía como objetivo a optimizar, posteriormente, resuelven el problema original. De forma complementaria, los segundos proponen una metaheurística de dos fases basada en el ICA y en VNS. Otros autores, no han considerado el *makespan* como objetivo y se han centrado en la tardanza y el consumo de energía. (T. Jiang & Deng, 2018) tienen en cuenta, la energía consumida por elementos independientes como aires acondicionados o televisores y desarrollan una Optimización por Enjambre de Gato Discreta – Bipoblación (BDCSO) para solucionar el problema. Por su parte, (R. Zhang & Chiong, 2016) plantean un problema con máquinas cuya velocidad de procesamiento es variable y un MOGA con dos estrategias de mejora local como solución.

También se destacan trabajos de investigación del problema enfocado a otros objetivos. (Lei et al., 2017) proponen el consumo de energía y la carga total de trabajo, además, plantean velocidades de procesamiento variables y un SFLA cómo metodología de solución. (Piroozfard et al., 2018) y (W. Liao & Wang, 2018) toman la huella de carbono y no el consumo de energía, como medida ambiental, aunque, el consumo se mide como parte de la huella. En este caso, los primeros proponen una investigación novedosa y pionera al incorporar el criterio total de trabajo tardío como objetivo a optimizar, solucionando el problema por medio de un MOGA. Y los segundos utilizan el *makespan* como objetivo, e incorporan el mantenimiento preventivo como variable de decisión, desarrollando también un MOGA como método de solución.

Los problemas multiobjetivo han sido estudiados y se adaptan mejor a los requerimientos de los sistemas productivos reales. Por esto, han adquirido gran relevancia en la actualidad. Para el tema en estudio, se encuentran investigaciones con incluso tres objetivos planteados, dos de ellos son el

makespan y el consumo de energía. Los siguientes autores presentaron en sus investigaciones, un FJSP en el que se incluyen tiempos de Setup y se propone un Algoritmo Genético Multiobjetivo (MOGA) para solucionar el problema. (Mokhtari & Hasani, 2017) tienen en cuenta en su planteamiento, variables referentes con el mantenimiento y la energía que es consumida por esta actividad, su tercer objetivo es la disponibilidad del sistema. Por su parte, (Salido, Escamilla & et. al., 2017) incluyen en su formulación un umbral de consumo de energía, máquinas con velocidades variables y consideración de los tiempos ociosos. El tercer objetivo de esta investigación es la robustez de la programación, entendida como: la capacidad que tiene el sistema de soportar la disrupción de una máquina, afectando, únicamente, el tiempo de terminación de la tarea en ejecución. De manera similar, (Yin, Li, Gao, et. al., 2017) formulan el problema con velocidades de procesamiento variables, añadiendo el factor ruido como objetivo a optimizar.

2.5 Condusiones

Como conclusión a este capítulo, las investigaciones analizadas hacen especial énfasis en la minimización de objetivos ambientales como el consumo energético y la huella de carbono, relacionándolos con objetivos como el *makespan* y la tardanza. Este enfoque en la investigación ha tomado relevancia debido a que no es suficiente con cumplir tiempos establecidos y dados al cliente, sino que, teniendo en cuenta las condiciones y exigencias actuales, resulta esencial demostrar que los procesos implementados tienen un impacto ambiental reducido.

Por lo anterior, se propone un método para minimizar y balancear los objetivos de tiempo de producción medido como *makespan* y de consumo energético en entornos de producción *Flexible Job Shop*. La perspectiva presentada en esta investigación aporta una metodología y una solución innovadora que abordan directamente este desafío, tanto en la implementación práctica de sistemas de manufactura como en la literatura académica. De esta manera, se contribuye con un enfoque nuevo y eficiente para abordar los problemas contemporáneos en el ámbito de la manufactura, cerrando la brecha entre la teoría y la aplicación.

3 Desarrollo del problema de programación de la producción en un ambiente de tipo FJS: Balanceo de makespan y consumo energético.

3.1 Introducción

El objetivo de este capítulo es proponer un modelo que solucione la problemática contextualizada en el capítulo anterior. Teniendo en cuenta todas las características deseadas para el sistema, en este capítulo se propone una alternativa de programación de producción que permite minimizar la medida de eficiencia (*makespan*) y la de eficacia (consumo energético) en entornos de manufactura *Flexible Job Shop* (FJS).

En la primera sección de este capítulo, el problema FJS con ahorro de energía se formula como un modelo matemático de optimización, por el cual se brinda una solución al problema de programación, para instancias pequeñas. Posteriormente, en la segunda sección del capítulo, se presenta el algoritmo metaheurístico, algoritmo genético de ordenamiento no dominado, desarrollado para la solución de instancias de gran tamaño que son más cercanas a las operaciones en entornos de manufactura reales.

Las alternativas de programación que se proponen en este capítulo fueron desarrolladas teniendo en cuenta lo indicado en los alcances y limitaciones expuestos en la sección 1.3 de este documento.

3.2 Representación matemática: solución exacta del problema de programación de producción, con minimización de makespan y consumo energético

En esta sección se presenta el problema de programación *Flexible Job Shop* el ámbito de la programación lineal entera mixta. El problema abordado en esta investigación sigue la definición clásica del FJS y además incorpora una función objetivo y una restricción adicional que no forman parte de la teoría tradicional. Esta restricción permite el balance y cumplimiento de los objetivos de eficiencia en la producción y consumo de energía.

A continuación, se presenta el modelo de Programación Lineal Entera Mixta (MILP: *Mixed Integer Linear Program*) para resolver este problema. Para la definición del modelo matemático de optimización, se tienen en cuenta los siguientes supuestos:

a) Todas las máquinas están disponibles en el momento t = 0.
b) Todos los trabajos están disponibles en el momento t = 0.
c) Cada operación puede ser procesada por una sola máquina al tiempo.

d) Los trabajos son independientes, es decir, no hay restricciones de precedencia entre las operaciones de los distintos trabajos.
e) No hay preferencia para las operaciones. Una vez una operación es iniciada no puede ser interrumpida.
f) Tiempo de transporte de un trabajo entre máquinas y tiempo de alistamiento de la máquina para el procesamiento de una operación está incluido en el tiempo de procesamiento.

Teniendo en cuenta lo anterior, a continuación, se describen los conjuntos, parámetros, variables de decisión, restricciones y funciones objetivo, que componen la formulación matemática propuesta para dar solución a instancias del problema FJS con dos objetivos.

3.2.1 Conjuntos

i	Trabajos	donde i $\in I$
j	Operaciones	donde j $\in J$
k	Máquinas	donde k $\in K$

3.2.2 Parámetros

$\mathbf{maq\_hab_{i,j,k}}$	Operación $O_{i,j}$ que puede ejecutar la máquina k (matriz binaria).
$\mathbf{proc\_op_{i,j,k}}$	Tiempo de procesamiento de la operación Oij en la máquina k [min]
$\mathbf{consu\_ene_k}$	Consumo energético de cada máquina k ($/min) o (kW/min) $[\frac{kW}{min}]$
L	Número muy grande

3.2.3 Variables de decisión

$\boldsymbol{Cmax}$	Makespan
$\boldsymbol{C_{ij}}$	Tiempo de finalización de la operación O_{ij}
$\boldsymbol{X_{ijk}}$	1 si la operación O_{ij} se procesa en la máquina k, 0 en caso contrario
$\boldsymbol{Y_{ijlwk}}$	1 si la operación O_{ij} se procesa antes que la operación O_{wl} en la máquina k, 0 en caso contrario
$\boldsymbol{S_{ij}}$	Tiempo de inicio de la operación O_{ij}

3.2.4 Funciones objetivo

Como funciones objetivo se plantean la minimización del *makespan* y del consumo energético.

$$\boldsymbol{Min\ Z1} = Cmax \quad (1)$$

$$\boldsymbol{Z2} = \sum_{\substack{i\in I\\ j\in J\\ k\in K}} X_{ijk} * proc\_op_{ijk} * consu\_ene_k + \sum_{\substack{i\in I\\ j\in J\\ k\in K}} Cmax - (X_{ijk} * proc\_op_{ijk}) * \frac{1}{4} consu\_ene_k \quad (2)$$

La ecuación (1) calcula el *makespan* y pretende su minimización. En la ecuación (2) se calcula el consumo energético como una suma de los tiempos activos y tiempos muertos de las máquinas y también se busca su minimización.

3.2.5 Restricciones

$$\forall_{k\in K}\, \forall_{i\in I, i\neq 0} \forall_{j\in J_i, j\neq 0}\ ;\ \sum_{\substack{l\in I\\ w\in J}} Y_{ijlwk} = Xijk \quad (3)$$

$$\forall_{k\in K}\ \forall_{i\in I, i\neq 0} \forall_{j\in J_i, j\neq 0}\ ;\ \sum_{\substack{l\in I\\ w\in J}} Y_{lwijk} = Xijk \quad (4)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{j\in J_i, j\neq 0};\ \sum_{k\in K} X_{ijk} = 1 \quad (5)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{j\in J_i, j\neq 0} \forall_{k\in K}\ ;\ X_{ijk} \leq maq\_hab_{ijk} \quad (6)$$

$$\forall_{k\in K}\ ;\ \sum_{\substack{i\in I\\ j\in J_i}} Y_{00ijk} = 1 \quad (7)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{j\in J_i, j\neq 0}\ ;\ C_{ij} = S_{ij} + \sum_k X_{ijk} * proc\_op_{ijk} \quad (8)$$

$$\forall_{i\in I}\, \forall_{\substack{j\in J_i\\ j>1}}\ ;\ S_{ij} \geq C_{i(j-1)} \quad (9)$$

$$\forall_{i\in I}\, \forall_{\substack{j\in J_i\\ j>1}}\ ;\ C_{00} = 0 \quad (10)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{j\in J_i, j\neq 0}\, \forall_{l\in I}\, \forall_{w\in J_l} \forall_{k\in K}, i\neq l\ ;\ S_{ij} \geq C_{lw} - \left(1 - Y_{lwijk}\right) * L \quad (11)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{j\in J_i, j\neq 0} \forall_{k\in K}\ ;\ Y_{ijijk} = 0 \quad (12)$$

$$\forall_{j\in J_i, j\neq 0}\, \forall_{l\in I}\, \forall_{w\in J} \forall_{k\in K}\ ;\ Y_{0jlwk} = 0 \quad (13)$$

$$\forall_{i\in I}\, \forall_{j\in J_i, j\neq 0} \forall_{k\in K};\ Y_{ij00k} = 0 \quad (14)$$

$$\forall_{i\in I}\, \forall_{j\in J_i}\, \forall_{w\in J, w\neq 0}\, \forall_{k\in K}\ ;\ Y_{ij0wk} = 0 \quad (15)$$

$$\forall_{i\in I, i\neq 0}\, \forall_{l\in I}\ \forall_{w\in J,} \forall_{k\in K}\ ;\ Y_{i0lwk} = 0 \quad (16)$$

$$\forall_{i\in I,} \forall_{l\in I, i\neq 0}\, \forall_{j\in J_i} \forall_{k\in K}\ ;\ Y_{ijl0k} = 0 \quad (17)$$

$$\forall_{i\in I}\, \forall_{j\in J}\ ;\ C_{max} \geq C_{ij} \quad (18)$$

$$\sum_{\substack{i\in I\\ j\in J\\ k\in K\\ q\in Q}} X_{ijk} * proc_{op_{ijk}} * consu_{ene_k} + \sum_{\substack{i\in I\\ j\in J\\ k\in K}} Cmax - (X_{ijk} * proc\_op_{ijk})) * \frac{1}{4} consu\_ene_k \geq 0 \quad (19)$$

$$\forall_{i\in I}\, \forall_{j\in J,} \forall_{k\in K};\ X_{ijk} \in \{0,1\} \quad (20)$$

$$\forall_{i\in I}\, \forall_{j\in J_i}\, \forall_{l\in I}\, \forall_{w\in J} \forall_{k\in K};\ Y_{ijlwk} \in \{0,1\} \quad (21)$$

$$\forall_{i\in I}\, \forall_{j\in J_i};\ C_{ij} \geq 0 \quad (22)$$

$$\forall_{i\in I}\, \forall_{j\in J_i};\ S_{ij} \geq 0 \quad (23)$$

$$C_{max} \geq 0 \quad (24)$$

En el conjunto de ecuaciones anteriores, se representan todas las características deseadas para el sistema estudiado. A continuación, se describe a detalle cada una de estas ecuaciones:

- Las ecuaciones (3) y (4) son ecuaciones de secuenciación. Con estas se asegura que cuando una operación está asignada a una máquina, a lo más, tenga una operación antecesora y una sucesora.
- Con la ecuación (5) se garantiza que cada operación j de un trabajo i se hace en una máquina k.
- La ecuación (6) determina las máquinas k capaces de procesar la operación j del trabajo i.
- Con la ecuación (7) se asegura que para cada máquina k hay solo una operación j que inicia la secuencia.
- Por medio de la ecuación (8) se calcula el tiempo de finalización de la operación j del trabajo i.
- Las ecuaciones (9) y (10) son ecuaciones de operaciones. Con ellas se asegura que una operación j no se inicie hasta que su antecesora j-1 haya finalizado.
- La ecuación (11) asegura que la operación j del trabajo i no inicie si la máquina k asignada no está disponible.
- En esta formulación se creó un trabajo *dummy* para controlar el inicio de la programación. Las ecuaciones 12, 13, 14, 15, 16 y 17 se crearon para facilitar este control. El trabajo *dummy* representa un trabajo que en la ejecución real no se realizará pero que ayuda a controlar el inicio de las operaciones y la factibilidad del modelo.
- La ecuación 18 calcula el valor del *makespan*.
- La ecuación 19 calcula el costo energético.
- Las ecuaciones 20 y 21 indican el carácter binario de las variables X_{ijl} y Y_{ijlwk}
- Las ecuaciones 22, 23 y 24 indican la no negatividad de las variables C_{ij}, S_{ij} y C_{max}.

3.3 Algoritmo metaheurístico propuesto para la minimización del makespan y consumo energético

En esta sección se presenta el algoritmo propuesto para resolver el problema propuesto. En este trabajo, el modelo presentado en la sección anterior no se resuelve de forma exacta dada la naturaleza *NP-hard* del problema. Es decir, se considera un problema complejo de optimización combinatoria y difícil (*hard*) en términos de complejidad computacional, de allí que se establezca solucionarlo por medio de métodos heurísticos.

Se presenta a continuación cada una de las operaciones del algoritmo propuesto.

3.3.1 Codificación y decodificación

Teniendo en cuenta la flexibilidad de la ruta de procesamiento de las operaciones en un entorno *Flexible Job Shop*, se propone una codificación de doble cadena. La primera cadena es un esquema de desempate que asigna un valor numérico positivo a todas las operaciones de la producción y asigna el orden en la secuencia de la programación en las operacionces para cada cromosoma. La segunda cadena, se basa en la codificación de máquina, su valor indica la máquina que se asigna a cada operación. La selección de la máquina toma no sólo su tiempo de procesamiento, sino también el consumo de energía de la máquina.

La codificación presentada no requiere un proceso de reparación en la cadena de asignación de máquinas, dado que cada gen del cromosoma alberga valores de máquina válidos. De igual forma, en la cadena de desempate, no se ejecuta un procedimiento de reparación. En el caso de que dos operaciones estén listas para su asignación con valores de desempate iguales, se opta por seleccionar una de manera aleatoria uniforme.

La propuesta para la codificación de los cromosomas resultantes se muestra en la Tabla 3.

Tabla 3. Codificación de cromosoma.

O_{11}	O_{21}	O_{31}	O_{12}	O_{22}	O_{32}
Cadena de operador de desempate					
3	*6*	*1*	*5*	*2*	*4*
Cadena de selección de máquinas					
M_2	M_3	M_2	M_3	M_1	M_2

3.3.2 Inicialización de la población

La población inicial N se genera de forma aleatoria de acuerdo con la codificación explicada en la sesión anterior. Los cromosomas de la población inicial son los padres de la primera generación del algoritmo. Para cada cromosoma de la población se evalúan las dos funciones objetivo que se buscan minimizar, *makespan* y consumo energético. A cada cromosoma se asocia un valor en cuanto a *makespan* F_1 y costo relacionado con el consumo energético de las máquinas F_2, estos valores en conjunto representan una solución al problema.

3.3.3 Función objetivo

La evaluación de los objetivos de cada cromosoma se realiza siguiendo el procedimiento descrito en Deb et al. (2002) y utilizando la metodología de ordenamiento elitista no dominado.

Evaluación de funciones

El objetivo de las funciones evaluadas es minimizar el *makespan* y minimizar el consumo energético de las máquinas. Para llegar a la solución se determina la secuencia de las operaciones en

las máquinas, la cual es factible si respeta las restricciones de precedencia entre las operaciones del mismo trabajo, es decir, la operación O_{j+1i} no puede ser procesada antes que la operación O_{ji}.

En el proceso de secuenciación, se identifican las operaciones *j* que están listas para ser asignadas a una máquina, una operación está lista si es la primera operación de un trabajo, es decir O_{1i} o si su operación predecesora ya ha sido ejecutada. Por lo general habrá más de una operación por ser asignada, por lo que el algoritmo asigna teniendo en cuenta las siguientes reglas:

1. Si hay dos o más operaciones esperando ser secuenciadas, se asignará primero la operación que tenga menor valor en la cadena de desempate.
2. Una vez se asigna la máquina, se completa el segundo componente del cromosoma con el valor de la máquina correspondiente y su consumo de energía respectivo.

El tiempo de inicio de procesamiento de una operación, dependerá de si la máquina asignada está ocupada o no. Cuando la máquina esté ocupada, el tiempo de inicio de procesamiento de la operación es inmediatamente después del término de la operación realizada previamente en la máquina; en caso contrario, el tiempo de inicio será inmediatamente se finalice la operación predecesora. Una vez todas las operaciones son asignadas y secuenciadas, se evalúan los objetivos, el primero el tiempo de finalización del último trabajo que deja el sistema, es decir el *makespan*, y en segundo lugar el costo del consumo energético final medido.

Ordenamiento

Cada solución obtenida a partir de los *N* cromosomas, se evalúa y se ordena de acuerdo con la no dominancia de sus valores en diferentes fronteras de no dominancia. La primera frontera se construye al comparar cada solución con todas las demás soluciones resultado de la población *N*, el fin de esto es encontrar las soluciones que no son dominadas por ninguna otra solución. Las soluciones no dominadas conforman una primera frontera F_1, para que posteriormente las soluciones restantes de la población *N* se comparen nuevamente entre sí y se conforme un nuevo conjunto de soluciones no dominadas F_2. Con esta técnica de ordenamiento, es claro que las soluciones que conforman la frontera F_2 no son dominadas por el conjunto de soluciones que quedan como remanentes en la población N, aun así, F_2 si es dominada por el conjunto de soluciones que conforman la frontera F_1. El proceso de creación de fronteras se repite sucesivamente hasta encontrar todas las fronteras F_n en las que se clasifican cada una de las soluciones de los cromosomas de la población N. A continuación, se presenta el seudocódigo (procedimiento) utilizado en la operación de ordenamiento no dominado.

$P' = \{1\}$
para cada $p \in P \wedge p \notin P'$
    $P' = P' \cup \{p\}$
    Para cada $q \in P' \wedge q \neq p$
    Si $p \prec q$, *entonces* $P' = P \setminus \{q\}$

Caso contrario, si $q \prec p$, *entonces* $P' = P' \setminus \{p\}$

El seudocódigo presenta un procedimiento en el que cada solución de la población P se comprueba con una población parcial de dominancia. La primera solución de la población se guarda en un conjunto P'. A continuación, cada solución p (la segunda solución en adelante) se compara con todos los miembros del conjunto P' uno a uno. Si la solución p domina a q de P', la solución q se elimina de P'. De este modo, se eliminan de P' los miembros no dominados de P'. En caso contrario, si la solución p está dominada por algún miembro de P', la solución p se ignora. Si la solución p no está dominada por ningún miembro de P', se introduce en P'. Considerando lo anterior, el conjunto P' crece con soluciones no dominadas. Cuando se validan todas las soluciones de la población, los miembros restantes de P' constituyen el conjunto no dominado y la primera frontera de Pareto. Finalmente, la dominancia en este trabajo se define como sigue:

$$f_{1B} < f_{1A} \wedge f_{2B} < f_{2A}$$

La anterior expresión implica que para que una solución B domine a una solución A, ambos objetivos (valores) de la solución B deben ser menores que los valores objetivos de la solución A. Si dicha condición no se cumple en ambos objetivos, no se tendrá dominancia de B sobre A. En la Figura 7 se describe el proceso de clasificación rápida de no dominancia (*fast non-dominated sorting*).

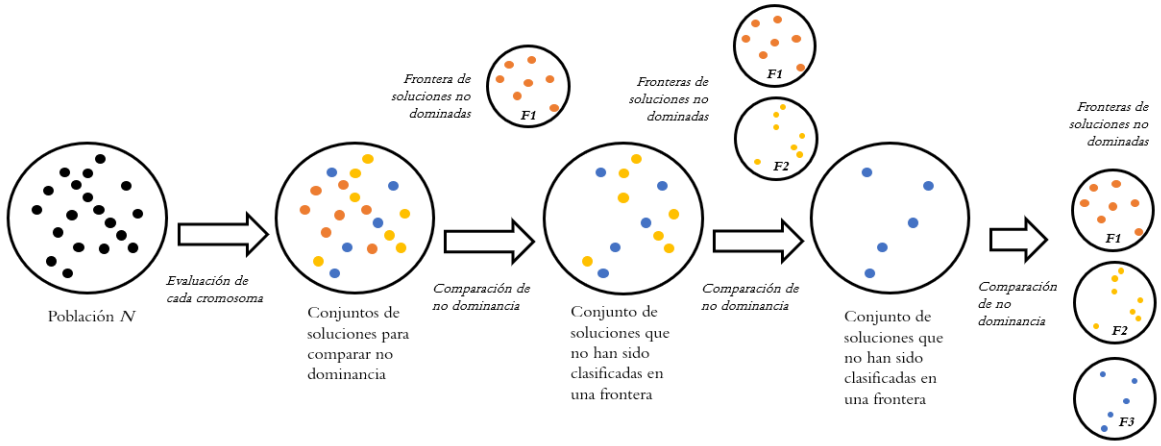


Figura 7. Descripción del proceso de clasificación rápida de no dominancia.

La clasificación de las soluciones de la población N en distintas fronteras no dominadas F, beneficia la convergencia del algoritmo hacia la curva óptima de Pareto. Sin embargo, es importante que la metaheurística permita encontrar una solución que presente una buena diversidad a lo largo del espacio de solución que abarca el Pareto (Talbi, 2009).

Estimación de la densidad

Por medio de la función de distancia de aglomeración (*crowding distance*) se calcula la distancia promedio de dos puntos a lo largo de cada uno de los objetivos. Con esta función se hace

una aproximación del perímetro del cuboide que se forma usando como vértices los vecinos más cercanos a la solución. Este proceso se ejecuta para garantizar la diversidad en las soluciones encontradas y para estimar la densidad de soluciones que rodean a un punto en concreto con cada uno de los puntos de la población que fueron clasificados en la misma frontera. En la Figura 8 se muestra el cuboide que se forma a partir del cálculo de la función de distancia de aglomeración de los puntos más cercanos a una solución *i*. Entre más grande sea el cuboide conformado por los puntos cercanos, mayor será la preferencia por la solución *i* dentro del conjunto existente de soluciones, ya que las mayores distancias benefician la diversidad de las soluciones que provee el algoritmo metaheurístico con respecto a la frontera de Pareto óptima. A continuación, se presenta el seudocódigo (procedimiento) utilizado en la operación de estimación de densidad.

l = |I|
para cada i, definir I|i| = 0
para cada objetivo m:
    I = ordenar(I, m)
    $I|1|_d = I|l|_d = inf$
    para i = 2 hasta (l – 1):
        $I|i|_d = I|i|_d + (I|i + 1|_{.m} - I|i - 1|_{.m})$

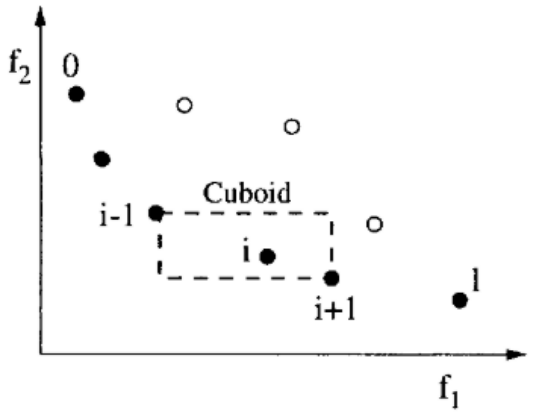


Figura 8. Cálculo de la función de distancia de aglomeración (Deb et al., 2002).

Teniendo en cuenta el procedimiento de ordenamiento no dominado y de estimación de densidad, se tiene que, en un conjunto de cromosomas de la población *N*, son preferibles las soluciones que se encuentren clasificadas en la frontera F_1, posteriormente, clasifican las que se encuentren en F_2 y así sucesivamente hasta llegar a la última frontera determinada. En caso de dos soluciones se encuentren clasificadas en una misma frontera, se prefieren aquellas que presenten menor densidad de puntos a su alrededor, es decir un valor de distancia de aglomeración mayor.

La aplicación de los seudocódigos presentados y descritos en esta sección se relaciona en el Apéndice B del presente documento.

Así, los cromosomas de toda la población N se ordenan, primero por el cálculo de la no dominancia que genera fronteras y posteriormente por la estimación de la densidad de los puntos pertenecientes a la solución.

3.3.4 Selección

La operación de selección se encarga de filtrar los cromosomas destinados a la reproducción. En la definición de este algoritmo, todos los cromosomas de la población inicial y de cada generación se eligen para formar parte de la piscina de reproducción. Sin embargo, se establece un criterio de ordenación lineal para organizar los cromosomas.

En lo que concierne a los cromosomas de la población inicial, estos son dispuestos en un orden ascendente basado en sus valores derivados de la función de evaluación según que se indica en la sección 3.3.3 del presente documento. Se tiene entonces que las soluciones que pertenecen a la frontera de Pareto F_1 se consideran las mejores hasta llegar a la última frontera. Dentro de las soluciones que pertenecen a cada frontera, se seleccionan primero aquellas con una distancia de aglomeración mayor. Posteriormente, se elige un par de cromosomas al azar, de forma uniforme y sin reemplazo (cada cromosoma solo puede ser seleccionado una vez) para la reproducción. La probabilidad de selección para cada cromosoma es equitativa, dado que se establece una distribución uniforme. Esta selección asegura la inclusión de individuos con distintas características, lo que amplía la diversidad en el proceso de reproducción. En otras palabras, se garantiza que, por ejemplo, un cromosoma de la frontera F_1 (indicativo de un buen individuo) pueda ser elegido para reproducirse con otro cromosoma que pertenezca a la frontera F_5 (indicativo de un individuo menos favorable), o incluso que dos "buenos individuos" puedan ser seleccionados para el entrecruzamiento.

Este procedimiento se repite hasta que se hayan seleccionado los pares de cromosomas correspondientes a la población inicial completa.

3.3.5 Entrecruzamiento

El propósito de esta etapa es mejorar de las soluciones a través de la transferencia de información entre los cromosomas elegidos para la reproducción. El procedimiento de cruzamiento consta de dos fases. En primer lugar, se toman los pares de cromosomas seleccionados en la fase previa (selección). En segundo lugar, se lleva a cabo el cruce en cada uno de estos pares, generando así dos nuevos cromosomas (hijos). Se opta por aplicar un operador de cruzamiento de múltiples puntos (denotado como pc), llevado a cabo en ambas cadenas del cromosoma.

Para ejemplificar, la Figura 9 muestra dos posibles cromosomas derivados de la codificación antes enunciada. Los dos cromosomas seleccionados como padres se someten al proceso de cruzamiento con dos puntos distintos, señalados mediante las flechas. En este proceso, se intercambia información de ambas cadenas.

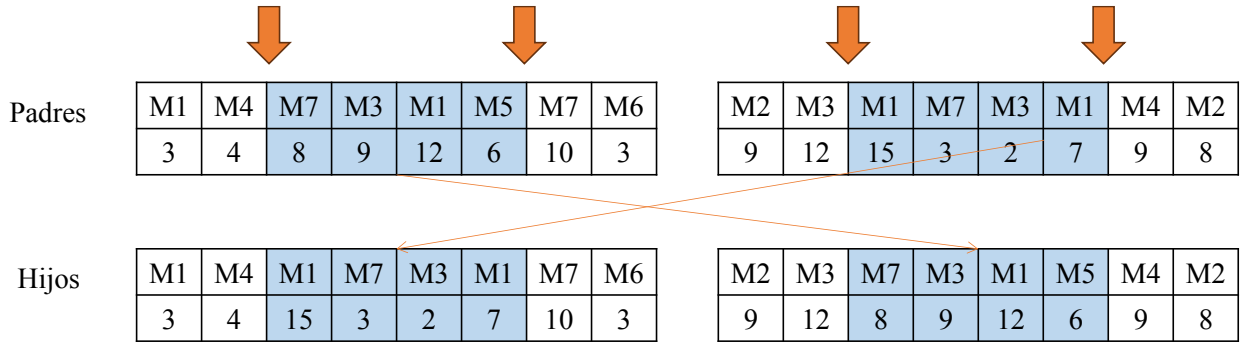


Figura 9. Entrecruzamiento del algoritmo genético propuesto.

3.3.6 Mutación

La operación de mutación seleccionada en este algoritmo solo afecta la cadena de selección de máquina. Esta operación funciona de la siguiente manera: se asocia un valor aleatorio (uniforme) de probabilidad a cada cromosoma, si este valor es menor a la tasa de mutación (*mr*) seleccionada, se realiza la mutación a dicho cromosoma. Se selecciona un gen del cromosoma a mutar de manera aleatoria uniforme. Después, se selecciona aleatoria y uniformemente una máquina del subconjunto de máquinas Rij ⊆ M, y se reasigna al gen anteriormente seleccionado. Este proceso es repetido para cada cromosoma de cada nueva generación obtenida.

3.3.7 Remplazo

Después de la producción de nuevos cromosomas (hijos) es necesario decidir qué cromosomas sobreviven en la nueva generación. Se implementó el siguiente procedimiento en este algoritmo: umbral de aceptación. Se define un punto de corte (*pc*) en la lista generada en la operación de selección (piscina de padres – población inicial). Este punto de corte es un valor porcentual sobre la longitud total de la lista (*N*). Los cromosomas de la piscina de reproducción ya ordenada (hijos) dentro del rango seleccionado hasta el punto de corte pasan a la siguiente generación. Para mantener la población total de cromosomas, *N*, pasan a la siguiente generación los cromosomas padres con mejor *makespan y consumo de energía*, hasta completar *N* individuos.

La Figura 10 muestra de manera general la operación de remplazo.

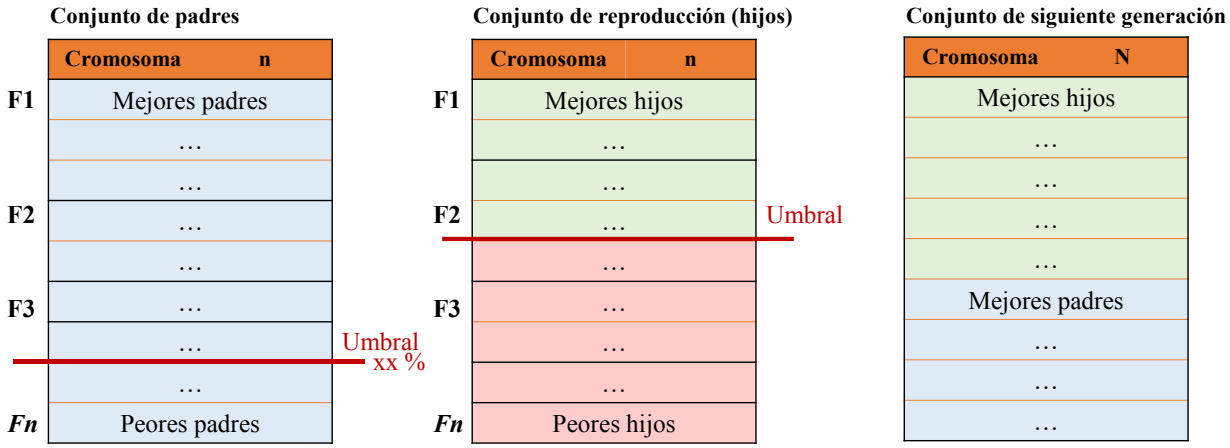


Figura 10. Remplazo en el algoritmo genético propuesto.

A continuación, se resumen los parámetros del algoritmo genético de ordenamiento no dominado:

- Tamaño de población (*N*): 1000.
- Número de generaciones (*ng*): 500.
- Puntos de entrecruzamiento (*pc*): 5.
- Probabilidad de mutación (*mr*): 2,5%.
- Umbral de aceptación (*thv*): 10%.
- Criterio de parada: el algoritmo se detiene si la cantidad de soluciones no únicas en una frontera óptima de Pareto supera 500 cromosomas o se ejecutan las generaciones definidas.

La Figura 11 muestra el diagrama de flujo del algoritmo NSGA-II planteado.

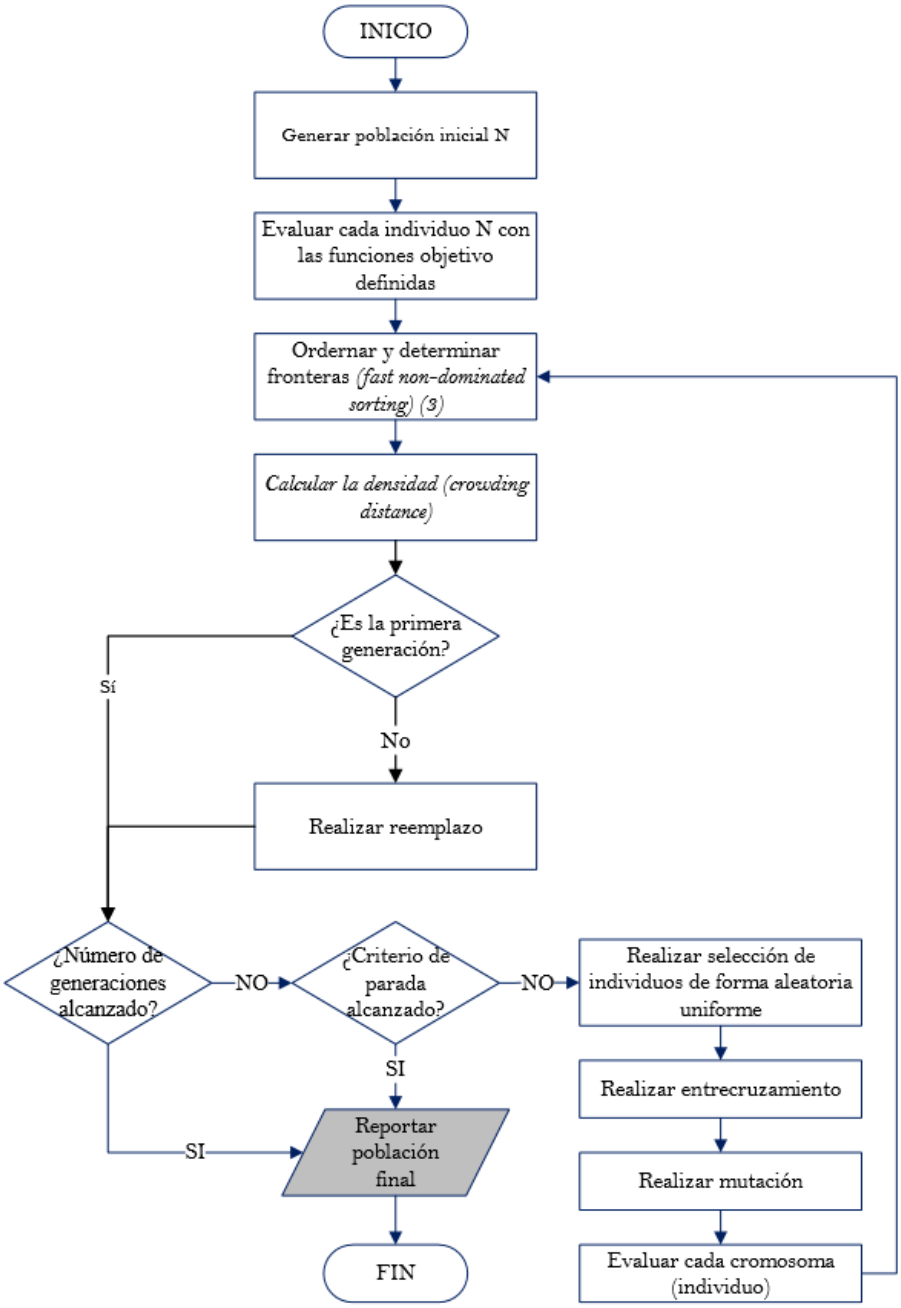


Figura 11. Diagrama de flujo algoritmo NSGA-II

4 Aplicación: caso de estudio experimental

4.1 Introducción

En este capítulo se implementa y validan los resultados del algoritmo genético de ordenamiento no dominado propuesto realizando experimentos en instancias reconocidas en la literatura como son las instancias de Brandimarte (1993) y Kacem (2002) .

La siguiente sección describe los detalles de las instancias. La segunda sección presenta el protocolo experimental que se lleva a cabo en la implementación funcional del algoritmo propuesto. En la tercera sección se muestra el detalle de la implementación del algoritmo con una instancia de ejemplo, se muestran los resultados obtenidos por la instancia definida siguiendo el protocolo experimental. Finalmente, se presentan los resultados de los experimentos realizados a cada una de las instancias presentadas.

4.2 Caso de estudio

El algoritmo presentado se aplicó a instancias de prueba de la literatura, particularmente se utilizaron las instancias de Brandimarte (1993) y Kacem (2002). Cada instancia está representada por una matriz *n x m*, donde *n* representa el número de máquinas y *m* es el número de trabajos el número total de operaciones y una flexibilidad que indica el número promedio de máquinas alternativas para cada operación de la instancia. La Tabla 4 presenta las instancias a considerar en el caso de estudio de este trabajo.

Tabla 4. Caracterización de las instancias de prueba.

Instancia	Configuración (*n x m*)	Operaciones totales	Flexibilidad
ka4x5	4 x 5	12	5
ka10x7	10 x 7	29	7
ka10x10	10 x 10	30	10
ka15x10	15 x 10	56	10
Mk01	10 x 6	55	2
Mk02	10 x 6	58	3.5
Mk03	15 x 8	150	3
Mk04	15 x 8	90	2
Mk05	15 x 4	106	1.5
Mk06	10 x 15	150	3
Mk07	20 x 5	100	3
Mk08	20 x 10	225	1.5
Mk09	20 x 10	240	3

Instancia	Configuración (*n x m*)	Operaciones totales	Flexibilidad
Mk10	20 x 15	240	3

Con el fin de considerar y evaluar el consumo de energía, este trabajo presenta una propuesta basada en los trabajos de Pach et al. (2015) y Seng et al. (2018). Se consideran tres estados posibles (*s*) en los que puede estar una maquina a saber: (i) máquina en reposo (*stand-by*), representa que la máquina no está trabajando, (ii) máquina en set-up, representa el inicio y finalización de la actividad de la máquina y (iii) máquina en operación, representa el tiempo efectivo de trabajo de la máquina. La Figura 12 representa los estados anteriormente descritos, en donde los tiempos notados como t_1 y t_6 no se consideran dentro de la operación de la máquina, el tiempo t_4 el estado *s(i)* máquina en reposo, los tiempos t_2 y t_6 representan el inicio y finalización de la operación de la máquina respectivamente (estado *s(ii)*) y los tiempos t_3 y t_5 representan tiempos de operación de la máquina notados por el estado *s(iii)*.

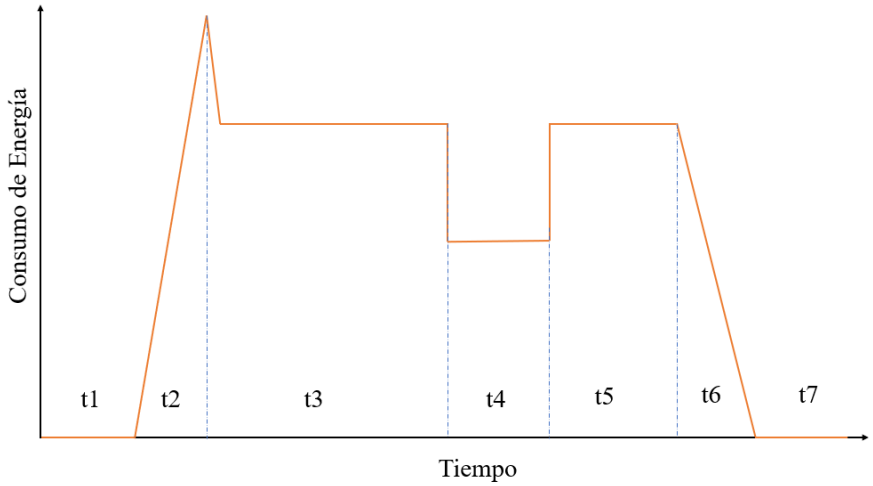


Figura 12. Estados para la máquina propuestos.

Adicionalmente, se consideran cuatro tipos de recursos o maquinas, las cuales se usarán en las instancias mencionadas como las máquinas que ejecutarán las operaciones, los detalles de la instanciación se explican en la sección *Protocolo experimental*. El consumo de energía para cada máquina considerando los tres estados mencionados se detalla en la Tabla 5.

Tabla 5. Consumo de energía de las máquinas.

Tipo de máquina (*R*)	Consumo de energía (W/unidad de tiempo)		
	Reposo	Inicio/fin (W/ciclo)	Operación
1	20	240	380
2	25	280	440
3	40	260	400

4	35	230	500

4.3 Protocolo experimental

Esta sección presenta el protocolo experimental para evaluar el algoritmo propuesto. La intención de realizar estos experimentos es evaluar el desempeño del algoritmo, medir el resultado sobre la medida inicial *makespan* y evaluar la capacidad de balancear el desempeño del sistema con el objetivo y medida de consumso de energía.

4.3.1 Experimentos

Los experimentos presentados usan un modelo de programación del algoritmo presentado implementado en Python con las instancias anteriormente descritas.

(1) Experimento A. No se considera el objetivo de consumo de energía. Se ejecuta el algoritmo únicamente para evaluar el desempeño del sistema con media de *makespan*. Este experimento se toma como línea base para validar los demás resultados y para configurar los parámetros del algoritmo.

(2) Experimento B. Considera el algoritmo multiobjetivo. Se ejecuta el algoritmo para evaluar ambos objetivos de desempeño en tiempo de ejecución del sistema y de consumo de energía a del sistema. Para validar el desempeño del algoritmo independientemente del consumo del recurso, se consideran únicamente dos tipos de recurso (máquina 2 y máquina 4).

(3) Experimento C. Considera el algoritmo multiobjetivo. Se ejecuta el algoritmo para evaluar ambos objetivos de desempeño en tiempo de ejecución del sistema y de consumo de energía a del sistema. Para validar el desempeño del algoritmo independientemente del consumo del recurso, se consideran tres tipos de recurso (máquina 1, máquina 2 y máquina 4).

(4) Experimento D. Considera el algoritmo multiobjetivo. Se ejecuta el algoritmo para evaluar ambos objetivos de desempeño en tiempo de ejecución del sistema y de consumo de energía a del sistema. Para validar el desempeño del algoritmo independientemente del consumo del recurso, se consideran los cuatro tipos de recurso propuestos.

4.3.2 Instancias

La Tabla 6 presenta el conjunto de instancias propuesto a partir de las instancias de la literatura mencionadas, los tipos de recursos presentados y los experimentos diseñados. La columna 'Tipo de recurso' indica la cantidad de tipos de recursos asignados a cada instancia considerando los recursos presentados en la Tabla 5.

Tabla 6. Instancias de experimentos.

ID	Instancia	*m x n*	Tipo de recursos (Cantidad de máquinas asignadas)								
			Experimento B		Experimento C			Experimento D			
			R_2	R_4	R_1	R_2	R_4	R_1	R_2	R_3	R_4
1	ka4x5	4 x 5	3	2	1	2	2	1	1	2	1
2	ka10x7	10 x 7	4	3	1	2	4	2	2	2	1
3	ka10x10	10 x 10	5	5	2	5	3	3	3	2	2
4	ka15x10	15 x 10	8	2	2	5	3	2	2	3	3
5	Mk01	10 x 6	3	3	1	2	3	1	2	2	1
6	Mk02	10 x 6	5	1	2	1	3	1	1	2	2
7	Mk03	15 x 8	4	4	2	3	3	2	1	3	2
8	Mk04	15 x 8	3	5	4	3	1	2	1	3	2
9	Mk05	15 x 4	2	2	2	1	1	1	1	1	1
10	Mk06	10 x 15	7	8	6	4	5	5	3	3	4
11	Mk07	20 x 5	2	3	3	1	1	1	2	1	1
12	Mk08	20 x 10	4	6	3	5	2	2	4	2	2
13	Mk09	20 x 10	5	5	5	2	3	3	2	3	2
14	Mk10	20 x 15	6	9	4	7	4	4	6	1	4

Para ilustrar la aplicación de las instancias presentadas, considérese la instancia ka4x5 mostrada en la Tabla 7 que consiste en 4 trabajos y 5 máquinas, dada la caracterización mostrada en la tabla anterior, se tendrán 3 máquinas del tipo recurso 2 (R_2) y dos máquinas del tipo recurso 4 (R_4). La asignación de cada maquina al recurso se hace en orden descendente, es decir, las maquinas M_1, M_2 y M_3 serán de tipo R_2 y las máquinas M_4 y M_5 serán de tipo R_4.

Tabla 7. Instancia ka4x5 (Kacem et al., 2002).

t		*M1*	*M2*	*M3*	*M4*	*M5*
J1	*O 11*	2	5	4	1	2
	O 12	5	4	5	7	5
	O 13	4	5	5	4	5
J2	*O 21*	2	5	4	7	8
	O 22	5	6	9	8	5
	O 23	4	5	4	54	5
J3	*O 31*	9	8	6	7	9
	O 32	6	1	2	5	4
	O 33	2	5	4	2	4
	O 34	4	5	2	1	5
J4	*O 41*	1	5	2	4	12

O 42	5	1	2	1	2

4.4 Implementación del algoritmo

En esta sección se muestra la implementación del algoritmo propuesto siguiendo los pasos descritos en el diagrama de flujo de la última sección del Capítulo 3 y relacionado en la Figura 11. Se indica el detalle de la implementación y se muestran los resultaos obtenidos para cada uno de los experimentos planteados para la instancia ka4x5.

El propósito general del algoritmo propuesto es minimizar los objetivos de tiempo de ejecución de la programación medido a través del indicador *makespan* y de consumo de energía considerando los tres estados propuestos para dicha medición. En consecuencia, el algoritmo genera las siguientes decisiones: (1) secuencia de entrada de los trabajos al sistema, (2) ruta de procesamiento de cada trabajo para completar sus operaciones, (3) momento de encendido y apagado de cada maquina y (4) tiempos de no operación de cada máquina.

4.4.1 Implementación técnica del caso de estudio experimental

El algoritmo y los experimentos fueron realizados en Python en un computador portátil Lenovo Yoga Slim 7 AMD Ryzen 7 4700U CPU@ 2,0 GHz con 16 GB de memoria RAM.

4.4.2 Instancia de implementación

Una vez se inicializa el algoritmo, se realiza la codificación del cromosoma, compuesta de dos cadenas como indicado en la sección 3.3.1. La Tabla 8 muestra la codificación de un cromosoma para la instancia seleccionada en esta implementación.

Tabla 8. Codificación de un cromosoma – ka4x5.

O_{11}	O_{12}	O_{13}	O_{21}	O_{22}	O_{23}	O_{31}	O_{32}	O_{33}	O_{34}	O_{41}	O_{42}
Cadena de operador de desempate											
8	*14*	*18*	*3*	*4*	*11*	*2*	*7*	*16*	*0*	*20*	*17*
Cadena de selección de máquinas											
M_1	M_2	M_5	M_2	M_4	M_3	M_4	M_3	M_5	M_4	M_1	M_2

Una vez se tiene la codificación, se procede a la evaluación de las funciones objetivos, para ello se procede con la secuenciación de operaciones siguiendo el procedimiento descrito en la sección 3.3.3. La Tabla 9 muestra un ejemplo del ciclo de asignación de operaciones teniendo en cuenta las dos cadenas de codificación anteriormente presentadas. Se tiene entonces que, la

operación O_{31} se asigna a la máquina M_4 ya su valor asignado en la cadena de desempate es 2 y resulta ser el menor con relación al de las actividades disponibles en el ciclo de asignación 1. Para la asignación 2, la operación O_{21} registra el menor valor en la cadena de desempate y se asigna en la máquina 2. Dicho procedimiento se realiza consecutivamente hasta realizar la asignación de la última operación.

Tabla 9. Ciclo de asignación de un cromosoma – ka4x5.

Operación	Máquina	Desempate
Cido de asignación 1		
O_{11}	M_1	8
O_{21}	M_2	3
O_{31}	M_4	2
O_{41}	M_1	20
Cido de asignación 2		
O_{11}	M_1	8
O_{21}	M_2	3
O_{32}	M_3	7
O_{41}	M_1	20
Cido de asignación 3		
O_{11}	M_1	8
O_{22}	M_4	4
O_{32}	M_3	7
O_{41}	M_1	20
Cido de asignación 4		
O_{11}	M_1	8
O_{23}	M_3	11
O_{32}	M_3	7
O_{41}	M_1	20
Cido de asignación 5		
O_{11}	M_1	8
O_{23}	M_3	11
O_{33}	M_5	16
O_{41}	M_1	20
Cido de asignación 6		
O_{12}	M_2	14
O_{23}	M_3	11
O_{33}	M_5	16
O_{41}	M_1	20

Operación	Máquina	Desempate
Cido de asignación 7		
O_{12}	M_2	14
O_{33}	M_5	16
O_{41}	M_1	20
Cido de asignación 8		
O_{13}	M_5	18
O_{33}	M_5	16
O_{41}	M_1	20
Cido de asignación 9		
O_{13}	M_5	18
O_{34}	M_4	0
O_{41}	M_1	20
Cido de asignación 10		
O_{13}	M_5	18
O_{41}	M_1	20
Cido de asignación 11		
O_{41}	M_1	20
Cido de asignación 12		
O_{42}	M_2	17

La Figura 13 muestra el diagrama Gantt correspondiente a la asignación del cromosoma ejemplo luego de la secuenciación y teniendo en cuenta los tiempos presentados en la Tabla 7. El *makespan* resultante es 19 unidades de tiempo. Adicionalmente, se muestra la cantidad de máquinas que operan en paralelo para todo el tiempo de producción. La grafica muestra el inicio de operación de tres máquinas (M_1, M_2 y M_4). De la misma manera, la finalización de la operación registra la salida de la máquina 5 en el tiempo 18 y de la maquina 3 en el tiempo 19.

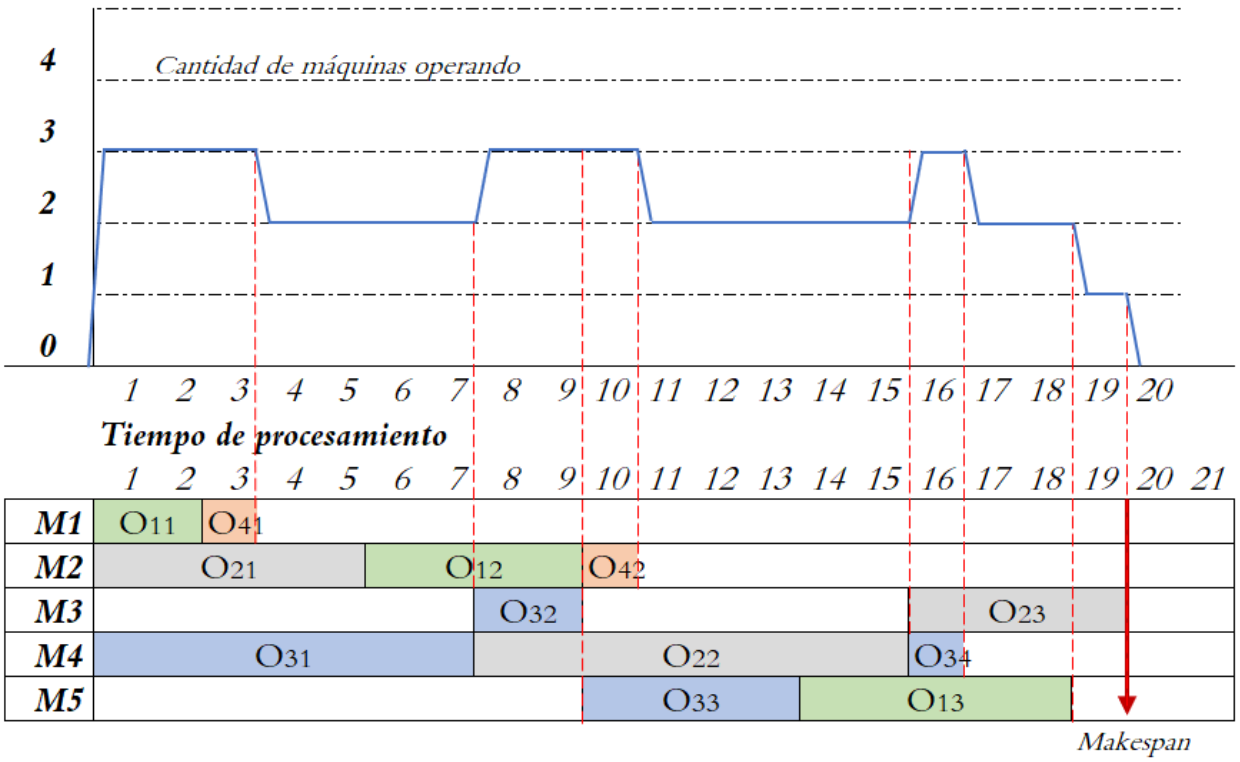


Figura 13. Diagrama de Gantt de cromosoma ejemplo - ka4x5.

La Tabla 10 presenta los costos de consumo de energía de acuerdo con la instanciación propuesta en la Tabla 5 y Tabla 6. El tipo de recurso se asigna de acuerdo con la instanciación y manteniendo el orden de las máquinas como indicado en la sección anterior. Para el cálculo de tiempo en reposo se consideran únicamente los tiempos muertos de la maquina durante el periodo de producción en operación. Dado lo anterior, la máquina 1 no presenta tiempos de reposo o muertos pues procesa en 3 unidades de tiempo y finaliza su producción. En este trabajo, el tiempo restante no se considera de reposo pues la máquina no está esperando otro trabajo. En el caso de la máquina 3, se considera tiempo de reposo el tiempo que transcurre entre la operación 2 del trabajo 3 (O_{32}) y la operación 3 del trabajo 2 (O_{23}). Se tienen 6 unidades de tiempo en reposo y estas se multiplican por los 25 W/unidad de tiempo que consumen las máquinas tipo recurso 2.

Tabla 10. Consumo energético para cada estado de máquina definido – ka4x5.

	Recurso	Reposo	Inicio/fin	Operación
M_1	R_2	-	280	1320
M_2	R_2	-	280	4440
M_3	R_2	150	560	2640
M_4	R_4	-	230	8000
M_5	R_4	-	230	4500

El cálculo del estado "inicio/fin" considera la cantidad de veces que se enciende/apaga una máquina y se multiplica por el consumo asociado. Para la máquina 1 se tiene un solo ciclo de encendido/apagado, dicho ciclo se multiplica por 280 W/ciclo asociado al recurso 2. Para la máquina 3, por ejemplo, se tienen 2 ciclos de encendido/apagado. Finalmente, El tiempo de operación de cada máquina, se multiplica por el consumo en W/unidad de tiempo para cada tipo de recurso. Por ejemplo, la maquina 4 opera 16 unidades de tiempo y es recurso tipo 4, por lo que su consumo es de 8000 W.

Una vez se secuencia un cromosoma, el procedimiento se repite para todos los cromosomas definidos y de dicha manera configurar la población inicial. El algoritmo continúa con la creación de las fronteras de Pareto, objetivo para el cual se apoya en el ordenamiento no dominado propuesto en la sección 3.3.3. La Figura 14 muestra un conjunto de fronteras de Pareto tras 10 generaciones del algoritmo para la instancia ka4x5. La frontera 1 (F1) en color azul muestra el conjunto de mejores soluciones o soluciones no dominadas por ningún otro conjunto de fronteras. En general, se puede validar la minimización de ambos objetivos, al encontrar la mejor frontera cercana al origen del plano. Asimismo, se valida la existencia de otras soluciones que son dominadas, como en el caso de la frontera 2 en naranja. Por ejemplo, el punto (14, 18430) que hace parte de la frontera 2, tiene el mismo valor de *makespan* (14) que el punto (14, 17965), sin embargo, no es menor en ambos objetivos por lo que no domina dicho punto y pasa a ser parte de la frontera 2 (ejemplo de dominancia). Se aclara, finalmente, que para tener un gráfico legible se incluyeron únicamente 10 fronteras ya que para dicha generación (10) se generaron 161 fronteras.

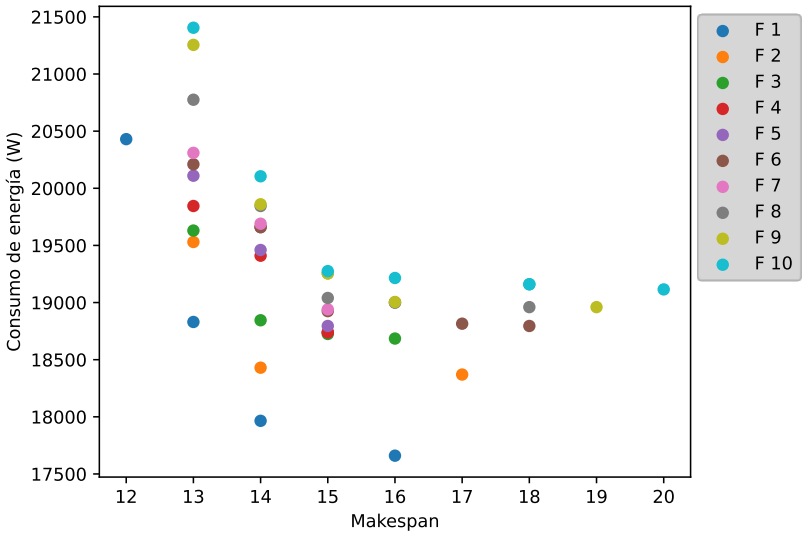


Figura 14. Fronteras de Pareto en la generación 10 – ka4x5.

La Figura 15 muestra las dos primeras fronteras de la mejor solución de la instancia ka4x5 generada por el algoritmo propuesto. En la figura hay una gran aglomeración de puntos superpuestos en la primera frontera, ya que la frontera se compone de 813 cromosomas de los 1000 en total, indicando que la diversidad en el espacio de solución es poca (tras 500 generaciones) y que el

algoritmo está apuntando a la misma solución como mejor solución. Asimismo, esta grafica permite ver que, hay diferentes soluciones posibles para escoger la que mejor se adapte a la situación de producción. Lo anterior implica que, aunque, el procedimiento de distancia de aglomeración prefiera soluciones con distancias más grandes, la variedad en las soluciones y el conocimiento de sus métricas teóricas puede soportar mejores procesos de toma de decisiones. Finalmente, la Figura 16 muestra las 10 primeras fronteras para la instancia confirmando el ordenamiento no dominado y las soluciones que se alejan de la minimización de los objetivos.

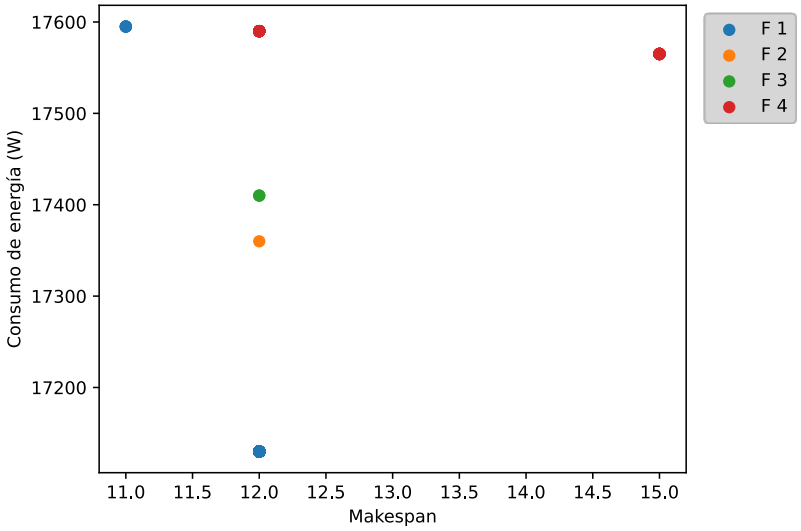


Figura 15. Primeras fronteras de Pareto en la mejor solución – ka4x5.

Para ejemplificar la distancia de aglomeración se considera el cuboide expuesto en la Figura 17. Se toma la solución señalada con rojo y que hace parte de la frontera 1 con objetivos (13, 15750). El cuboide se forma con las soluciones de la misma frontera inmediatamente siguientes, tanto hacia adelante como hacia atrás, de acuerdo con el orden generado y cuyos valores son (12, 20480) y (14, 17960). La distancia de aglomeración se calcula, como fue indicado en la sección 3.3.3, es decir, se realiza la resta de los objetivos que rodean la solución analizada y posteriormente se suman o lo que es lo mismo, se suman los lados del cuboide generado. El valor 2 mostrado en la figura es resultado de 14 – 12 y el valor 2520 es resultado de 20480 – 17960. Esta figura permite ver nuevamente la riqueza de soluciones y la posibilidad de toma decisiones dirigidas a un balance de la minimización de los objetivos o a una preferencia hacia alguno de ellos.

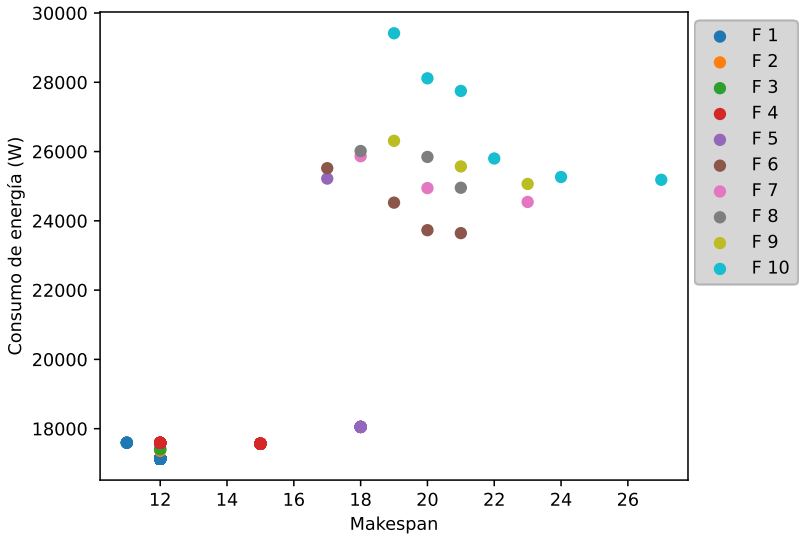


Figura 16. Fronteras de Pareto en la mejor solución – ka4x5.

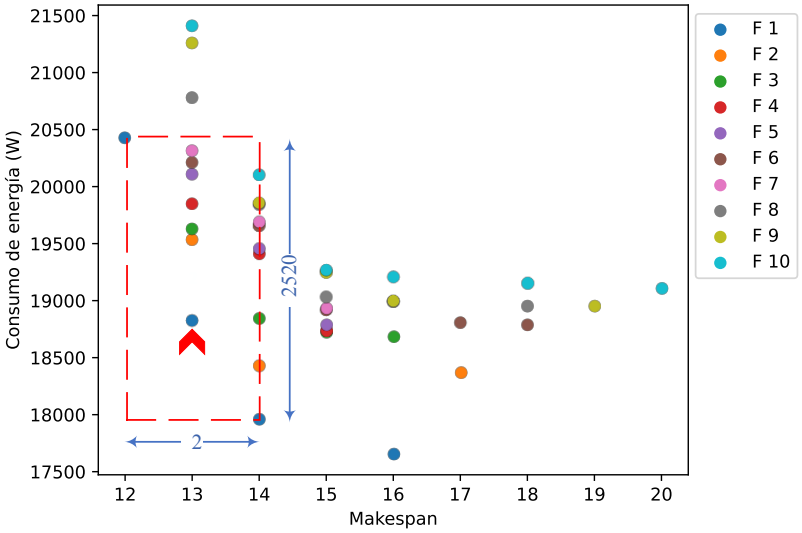


Figura 17. Distancia de aglomeración de una solución – ka4x5.

Los diagramas de Gantt y su correspondiente consumo de energía de las mejores soluciones que hacen parte de la primera frontera tras 500 generaciones se muestran a continuación. La Figura 18 muestra la solución para la instancia de implementación con un *makespan* de 14 y un consumo de energía de 17270 W. Esta solución muestra una preferencia por realizar la producción únicamente con cuatro maquinas, evitando así el consumo de una de ellas. Esto es posible dada la flexibilidad total de esta instancia, es decir, todas las máquinas son capaces de realizar todas las operaciones. Así mismo, la figura permite ver la cantidad de ciclos de inicio/fin, la cantidad de máquinas en operación al mismo tiempo y el tiempo de operación de cada máquina. En este apartado es importante resaltar que el estado de cada máquina es considerado fijo y con el tiempo que cada máquina permanece en un estado determinado es posible calcular el consumo total de energía.

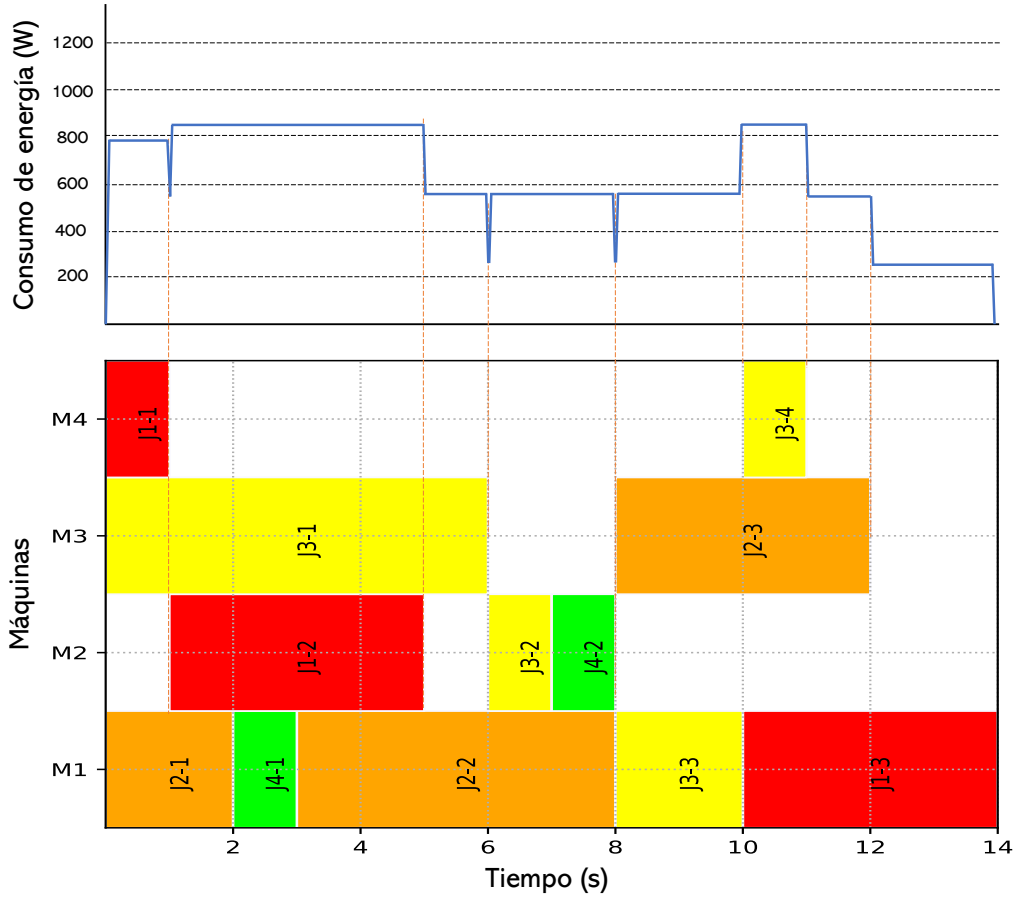


Figura 18. Diagrama de Gantt y consumo de energía para solución 1 ka4x5.

La Figura 19 muestra la solución para la instancia de implementación con un *makespan* de 12, representando una disminución porcentual de 14,28% con relación a la solución anterior y un consumo de energía de 17310 W que representa un aumento porcentual de 0,3% con relación a la solución anterior. Esta solución también prioriza el uso de cuatro máquinas y aunque disminuye el tiempo total de la ejecución de la producción aumenta el consumo de energía. Particularmente, la operación 3 del trabajo 3 (O_{33}) se traslada de la maquina 1 a la maquina 4 (con relación a la solución anterior) y aumenta el tiempo de operación de la maquina 4 generando el aumento global de consumo de energía.

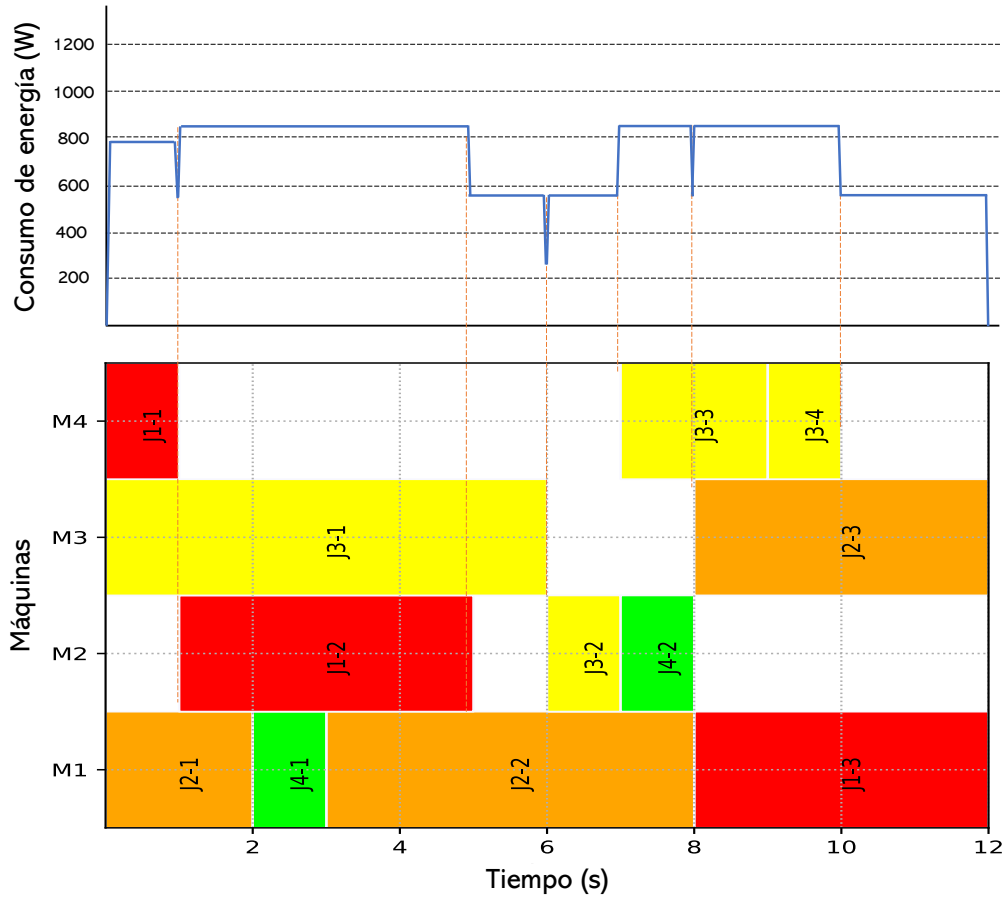


Figura 19. Diagrama de Gantt y consumo de energía para solución 2 ka4x5.

La Figura 20 muestra la solución para la instancia de implementación con un *makespan* de 11 y un consumo de energía de 17595 W. Esta solución presenta el mejor valor de *makespan*, pero un consumo de energía mayor con relación a las dos soluciones presentadas anteriormente. La grafica de consumo de energía muestra una utilización constante de las máquinas y de allí el mayor consumo energético de estas. Asimismo, por los valores de consumo de energía asignados a los estados de reposo e inicio/fin se añade una penalización que también tiene un impacto en el consumo de energía total del sistema.

La Tabla 11 resume los resultados obtenidos para los tres experimentos de la instancia utilizada en esta sección ka4x5, donde mkp hace referencia al valor de *makespan* y CE al valor de consumo de energía. Se muestran las soluciones que hacen parte de la mejor frontera de Pareto y por ello, algunos experimentos muestran más soluciones que otras. El experimento A, es decir, el experimento que no considera el objetivo de consumo de energía da como resultado un *makespan* de 11. La tabla permite ver como el algoritmo crea fronteras de Pareto eficientes en las que incluye soluciones con ambos objetivos minimizados y soluciones con valores "intermedios" entro los extremos de cada frontera. Lo último es de importancia ya que permite validar el desempeño del algoritmo y en la ejecución de la producción brinda alternativas de programación a los objetivos planteados de forma tal que se puede priorizar un objetivo o el otro, como se planteó en uno de los

objetivos de este trabajo, encontrar un balance entre ambos objetivos. Esto, como mencionado a través de las soluciones presentes en la mejor frontera de Pareto.

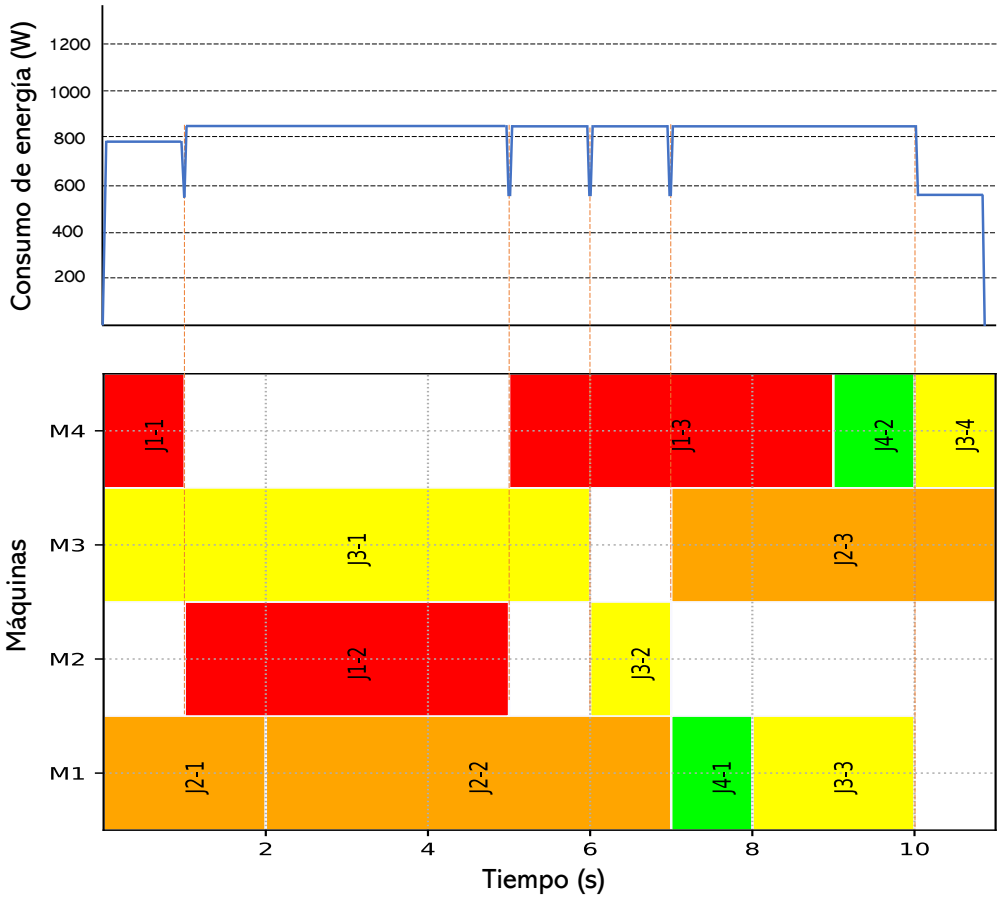


Figura 20. Diagrama de Gantt y consumo de energía para solución 3 ka4x5.

La Figura 21 muestra los resultados de los experimentos para la instancia antes enunciada. Esta figura muestra una tendencia para cada experimento en la que a medida que aumenta el *makespan* disminuye el consumo de energía para el sistema. Se puede identificar una relación lineal en la que el *makespan* se comporta como una variable independiente del consumo de energía (variable dependiente). Son de particular interés, para trabajos futuros que puedan surgir de esta investigación, los comportamientos de las rectas mencionadas, ya que estas representan casos genéricos de posibles entornos de manufactura. El comportamiento de estas rectas puede servir para validar los resultados obtenidos por otros investigadores.

Tabla 11. Resultados para la instancia ka4x5. Todos los experimentos.

		Experimento A		Experimento B		Experimento C		Experimento D	
		mkp (s)	CE (W)	mkp (s)	CE (W)	mkp (s)	CE (W)	mkp (s)	CE (W)
ka4x5	Solución 1	11	18560	14	17270	18	16170	13	15695
	Solución 2			12	17310	16	16310	12	15755

Solución 3	11	17595	14	16390	11	15895
Solución 4			12	16510		
Solución 5			11	16910		

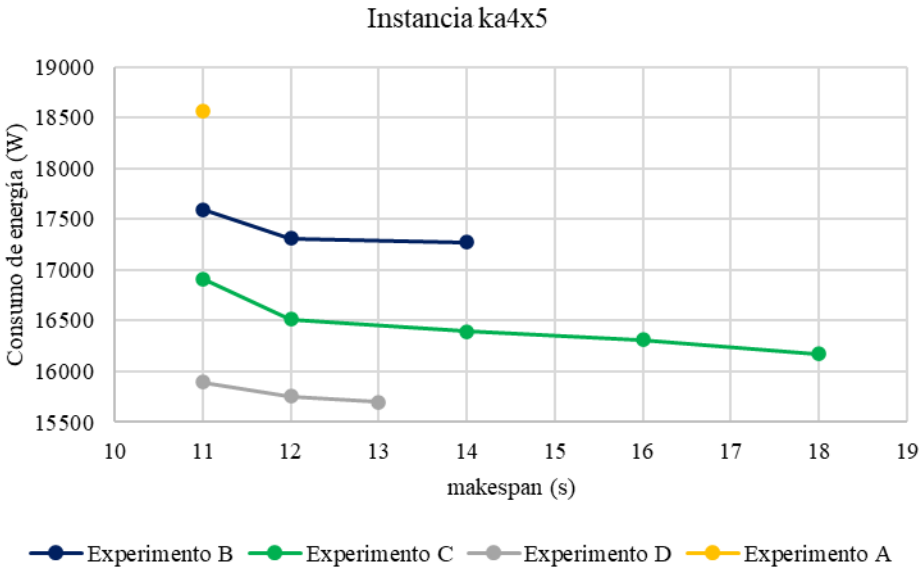


Figura 21. Resultados para la instancia ka4x5. Todos los experimentos.

4.5 Resultados

En esta sección se presentan los resultados de los experimentos realizados a cada una de las instancias presentadas en la Tabla 6.

La Tabla 12 presenta el valor de *makespan* y del consumo de energía obtenido el ejecutar el experimento A y B, donde mkp hace referencia al valor de *makespan* en segundos (s) y CE al valor de consumo de energía en vatios (W). Los resultados obtenidos en el experimento A solo consideran el objetivo de *makespan* y dicho experimento se usó para definir los parámetros del algoritmo a través de un diseño de experimentos el cual puede ser encontrado en el Apéndice A del presente documento. Estos resultados refuerzan la necesidad de resolución de la programación de operaciones en entornos de manufactura con diferentes objetivos, en este caso, minimizando tanto el tiempo de producción como el consumo de energía. Por ende, si la técnica de resolución no incorpora (o no está en capacidad de incorporar) todos los objetivos que se quieren considerar, las medidas de desempeño no evaluadas sufrirán una degradación, en la ejecución real, con respecto a lo obtenido de manera teórica. Al igual que en la Tabla 11, se muestran las soluciones de cada instancia que hacen parte de la mejor frontera de Pareto. Las últimas dos columnas muestran la variación porcentual del *makespan* y el consumo de energía, respectivamente, entre los valores obtenidos del experimento A y la mejor solución con relación al indicador de consumo de energía (Solución 1).

En general, se identifica una degradación en la medida de *makespan* de alrededor del 20% y una disminución en la medida de consumo de energía cercana al 5%.

Tabla 12. Resultados generales de los experimentos A y B.

Instancias (ID)	Experimentos													
	A		B											
	mkp	CE	Solución 1		Solución 2		Solución 3		Solución 4		Solución 5			
			mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp %	CE %
1	11	18560	14	17270	12	17310	11	17595					27.27%	-6.95%
2	11	33160	13	30735	12	31320							18.18%	-7.31%
3	7	24050	10	23265	9	23295	8	23555					42.86%	-3.26%
4	12	49270	18	48220	16	48570	15	48715					50.00%	-2.13%
5	40	80230	47	77105	46	77155	45	78185					17.50%	-3.90%
6	28	74890	37	69160	36	69260	33	69660					32.14%	-7.65%
7	204	459875	222	442665	213	442765							8.82%	-3.74%
8	65	203475	85	174195	80	175330	79	175590	75	177855	74	178475	30.77%	-14.39%
9	176	332640	183	325145									3.98%	-2.25%
10	75	226560	98	216260	96	216460	95	217080	94	222070			30.67%	-4.55%
11	144	341895	174	322275	167	323365	166	324670	156	327890	152	329225	20.83%	-5.74%
12	523	1312855	554	1311425	552	1311645	553	1311625					5.93%	-0.11%
13	345	1207455	422	1145425	414	1146105	412	1150115	408	1151685	407	1163205	22.32%	-5.14%
14	277	1097790	342	1043425	337	1044370	332	1045170	328	1047650			23.47%	-4.95%
												Promedio	23.91%	-5.15%

La Tabla 13 resume los resultados del Experimento C. Estos resultados refuerzan que, las técnicas de scheduling tradicionales o mono-objetivo no enfrentan de manera adecuada las características de los entornos de manufactura reales. Por ende, no están en capacidad de asegurar la optimalidad de medidas de desempeño no definidas.

Se refuerza también que, una función objetivo simple es una causa del desbalance entre los diferentes objetivos y medidas asociadas en problemas y entornos de programación de operaciones. En este caso, en las instancias evaluadas, tras la obtención de la mejor frontera de Pareto, las medidas de *makespan* y consumo de energía (esta última no fue optimizada en el experimento A) sufren variaciones porcentuales con relación al experimento A. La medida de *makespan* en algunas instancias es igual al valor optimo y la degradación promedio es cercana al 20%. La medida de consumo de energía se mejora en todos los resultados de la frontera de Pareto y es mejorada en la Solución 1 alrededor de un 5%.

Tabla 13. Resultados generales de los experimentos A y C.

Instancias (ID)	Experimentos													
	A		C											
	mkp	CE	Solución 1		Solución 2		Solución 3		Solución 4		Solución 5			
			mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp %	CE %
1	11	17955	18	16170	16	16310	12	16370	11	16910			63.64%	-9.94%
2	11	33160	13	30875	12	31405							18.18%	-6.89%
3	7	24050	10	22265	9	22275	8	22800	7	24130			42.86%	-7.42%
4	12	49270	17	46510	16	46575	15	46820	14	48755			41.67%	-5.60%
5	40	80230	47	76155	46	76210	45	77245					17.50%	-5.08%
6	28	74890	41	68685	40	68940	37	69260	35	69515	34	69575	46.43%	-8.29%
7	204	459875	249	414215	240	414900	231	416335	229	41380	222	422790	22.06%	-9.93%
8	65	203475	92	150785	86	152070	82	153630	80	153920	75	156365	41.54%	-25.90%
9	176	332640	186	293615	185	293620	184	294165	183	294275	178	295365	5.68%	-11.73%
10	75	226560	103	192500	101	192610	96	194350	95	194555	94	195150	37.33%	-15.03%
11	144	341895	162	283915	159	284320	150	284855					12.50%	-16.96%
12	523	1312855	578	1195270	571	1195910	568	1197230					10.52%	-8.96%
13	345	1207455	443	1041250	440	1041905	439	1043405	437	1044455	435	1045675	28.41%	-13.76%
14	277	1097790	332	932995	330	933455							19.86%	-15.01%
												Promedio	29.15%	-11.46%

La Tabla 14 resumen los resultados de los experimentos A y D. Estos resultados refuerzan que una inclusión de diferentes objetivos en técnicas metaheurísticas y en particular en un algoritmo genético, es un enfoque valido para balancear la consecución de objetivos definidos en un entorno de manufactura. Lo anterior se justifica en que, los resultados de los tres experimentos, por medio del algoritmo propuesto, aunque afecta negativamente el valor medido de *makespan*, reduce la degradación de esta medida al mismo tiempo que reduce el consumo de energía en el sistema de manufactura. Adicionalmente, las soluciones presentes en la frontera de Pareto reducen la medición de ambos objetivos, pero también refleja soluciones en las que se balancean ambas medidas de desempeño. Es decir, las medidas de desempeño "extremas" (Solución 1 y Solución 5) son las que muestran la mayor minimización de cada medida y las demás soluciones muestran el balance entre ambos objetivos.

A pesar de lo anterior, se debe tener en cuenta que, aunque los modelos tradicionales permiten la incorporación de funciones multiobjetivo, la elección de las diferentes medidas de desempeño debe ser un proceso riguroso (representación adecuada del sistema), pues un mayor número de objetivos implica una mayor dificultad en términos de solución computacional.

Tabla 14. Resultados generales de los experimentos A y D.

Instancias (ID)	Experimentos													
	A		D											
	mkp	CE	Solución 1		Solución 2		Solución 3		Solución 4		Solución 5			
			mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp	CE	mkp %	CE %
1	11	18560	12	15555	11	15895							9.09%	-16.19%
2	11	33160	13	28925	12	29220							18.18%	-12.77%
3	7	24050	8	21420	7	23015							14.29%	-10.94%
4	12	49270	16	46370	15	47195							33.33%	-5.89%
5	40	80230	46	73025	45	73580	44	73675					15.00%	-8.98%
6	28	74890	34	66910	33	67010	32	67800	31	68360			21.43%	-10.66%
7	204	459875	235	406080	234	406820	226	407625	225	407650			15.20%	-11.70%
8	65	203475	90	158895	84	159340	72	161780	70	163615	68	164625	38.46%	-21.91%
9	176	332640	183	296405									3.98%	-10.89%
10	75	226560	93	200530	94	199235	95	198880	96	197585			24.00%	-11.49%
11	144	341895	175	290870	173	291675	172	291960	166	292465	160	293990	21.53%	-14.92%
12	523	1312855	577	1198140	576	1199090	574	1199655					10.33%	-8.74%
13	345	1207455	462	1054135	457	1055705	452	1059980	451	1068550	446	1068595	33.91%	-12.70%
14	277	1097790	351	967170	346	967600	344	968570	342	968145			26.71%	-11.90%
												Promedio	20.39%	-12.12%

La Figura 22 muestra las soluciones en las primeras 10 fronteras de Pareto para las instancias de prueba en la ejecución del experimento D. Esta figura y la Tabla 14 muestran la desviación de las soluciones con relación al *makespan* optimo obtenido por el algoritmo, así como las mejoras obtenidas cuando se evalúa el objetivo de consumo de energía. En la mayoría de los casos se obtienen al menos dos soluciones optimas, brindando diferentes opciones en la toma de decisiones y para los casos donde hay más de dos soluciones, se puede encontrar un mejor balance entre el tiempo de ejecución de la programación y el consumo de energía. Por ejemplo, la instancia Mk02 reporta seis soluciones optimas (la Tabla 14 solo reporta cinco) asegurando diversidad en las programaciones que pueden ser ejecutadas.

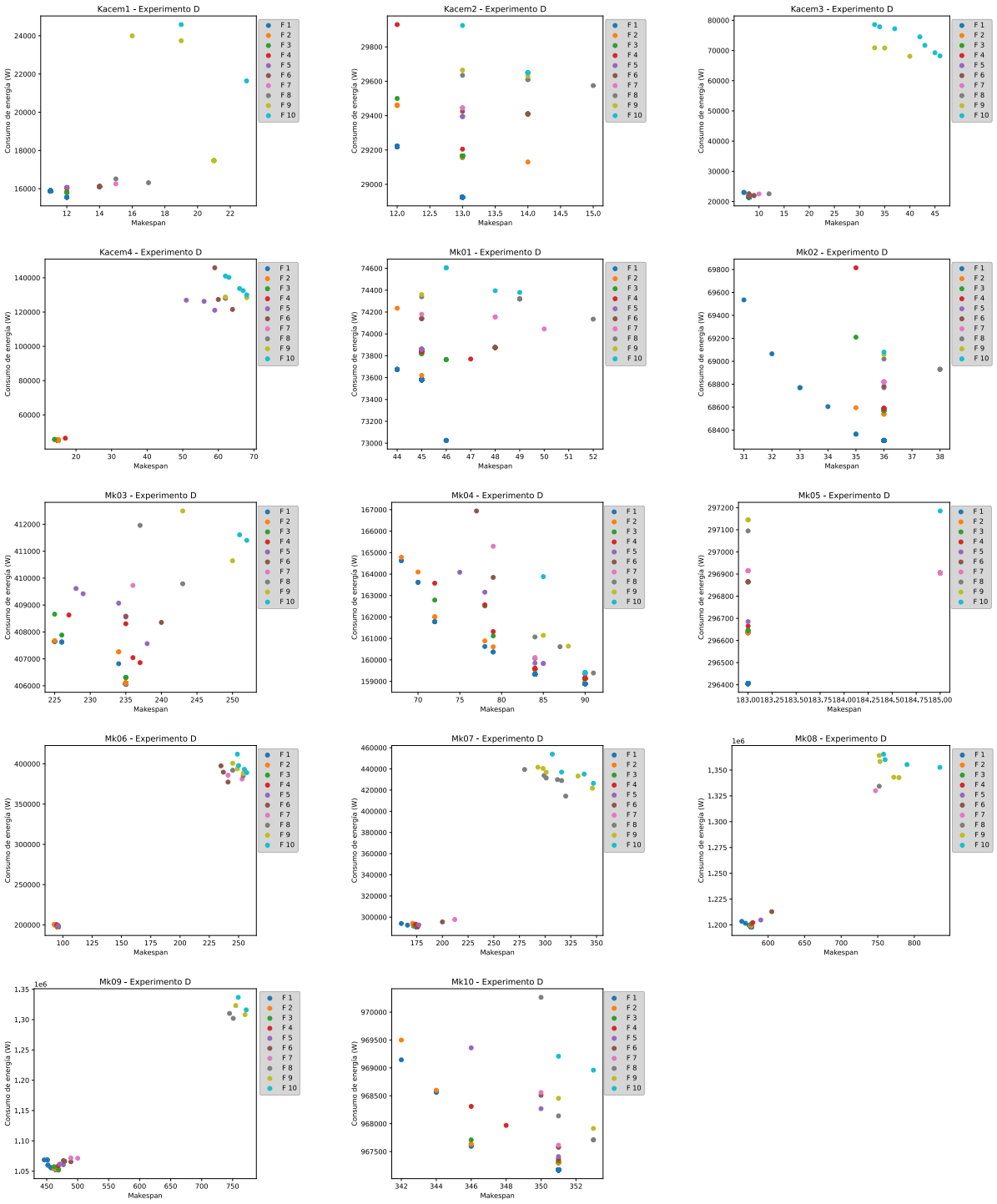


Figura 22. Fronteras de Pareto para las instancias propuestas. Experimento D.

En general, los resultados de los experimentos B, C y D muestran una reducción del consumo de energía con relación al experimento A de entre el 5 % y 12 %. Dicha mejora en la medida de consumo de energía implica una desmejora de alrededor del 20 % en el tiempo de ejecución total de la producción. Esto último, siendo válido para los escenarios en los que se

considera la mejor solución de consumo de energía, es decir, si se compara porcentualmente la mejor solución de la medida de *makespan* de los experimentos B, C y D la diferencia porcentual se disminuye a un 7 % aproximadamente en comparación con el experimento A.

Figura 23 muestra las fronteras de Pareto para la misma instancia, Kacem 3 (ka10x10), tras ejecutar los experimentos B, C y D. Esta figura permite ver que las soluciones que se acercan al mejor valor de *makespan* también brindan reducción en el indicador de consumo de energía. Por ejemplo, en el experimento B en el que hay una variedad de soluciones con diferentes valores de consumos de energía con las que se puede alcanzar el mismo valor de *makespan*, 9 s en este caso. Esto muestra que, aunque los tiempos de producción se puedan cumplir de acuerdo con lo establecido, la forma como se secuencian los productos y se ejecutan las opciones con las maquinas son de vital importancia para la consecución de un consumo de energía deseado, ya sea el mínimo o algún valor de referencia. La Figura 24 muestra los diagramas de Gantt para la instancia ka10x10 con una solución de *makespan* de 9 s, pero el diagrama de la izquierda con un valor de consumo de energía de 23775 W (mejor óptimo) y a la derecha con un valor de consumo de 24365 W. Por ejemplo, la máquina 10 tiene 2 ciclos de inicio/fin en la configuración derecha en contraste con un solo ciclo de inicio con la configuración de la izquierda. Asimismo, el tiempo de espera de maquina es mayor y en consecuencia su consumo. Además, la configuración de la izquierda emplea únicamente 9 de las 10 máquinas disponibles, a diferencia de la configuración de la derecha que emplea las 10 máquinas (todas) para la ejecución de la producción.

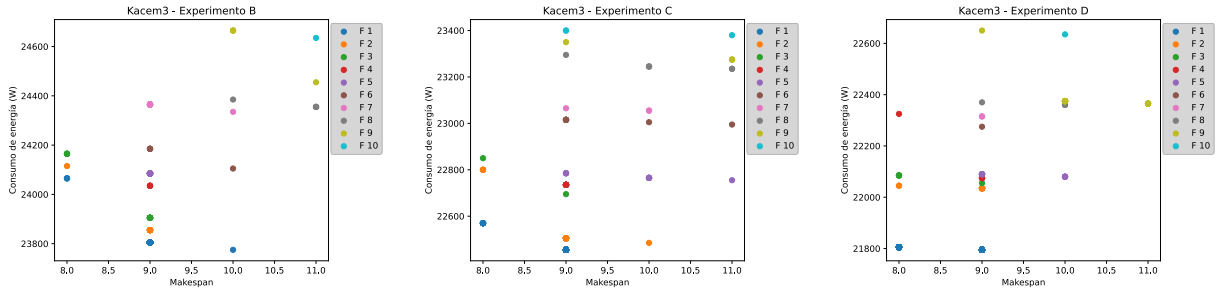


Figura 23. Fronteras de Pareto para instancia ka10x10. Todos los experimentos.

En la Figura 23 se puede apreciar también que el experimento D muestra un mismo valor de consumo de energía (21795 W) para dos valores diferentes de *makespan* (8 y 9 s respectivamente).

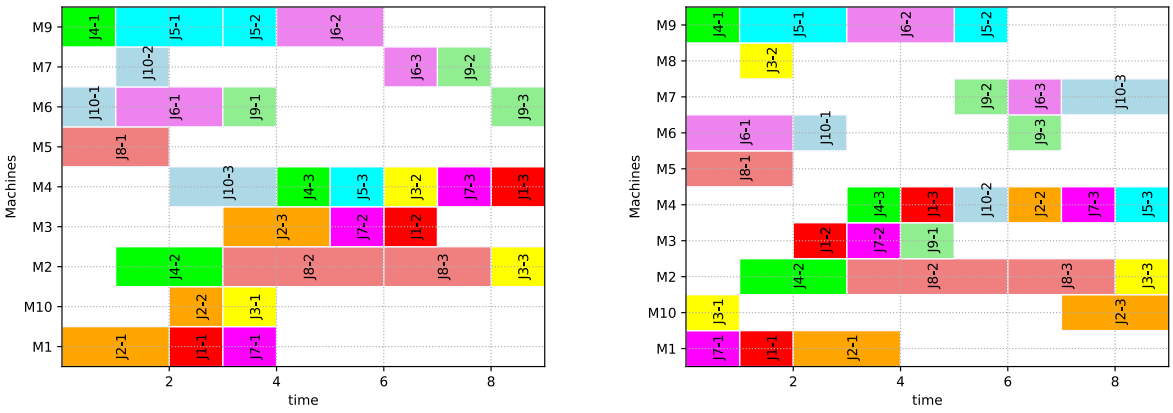
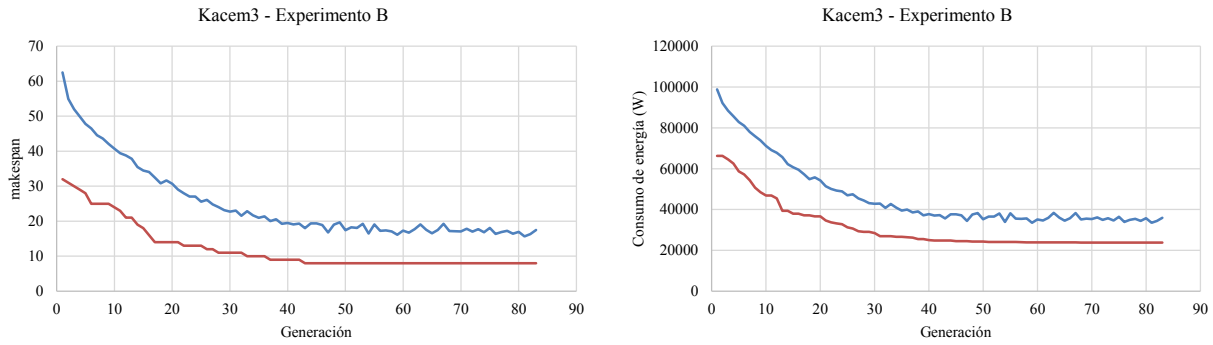


Figura 24. Diagramas de Gantt para instancia ka10x10.

Finalmente, se presentan las gráficas del desempeño del algoritmo para la instancia ka10x10 a modo de ejemplo. La Figura 25 muestra el desempeño del indicador *makespan* en la izquierda del gráfico y a la derecha se muestra el desempeño de la medición del consumo de energía, la línea azul representa el promedio de las corridas ejecutadas, mientras que la línea roja representa el comportamiento de la mejor solución. La figura permite ver la convergencia del algoritmo hacia ambos objetivos manteniendo la diversidad de la población y la intensificación de la solución tras la mejora en cada generación. Para los diferentes experimentos se logran minimizar ambos objetivos y esto es un valor para resaltar del algoritmo, pues se demuestra que indistintamente de la caracterización del sistema de manufactura, el algoritmo está en capacidad de lograr minimizar y balancear ambos objetivos. El algoritmo para el experimento B, alcanzó el criterio de parada tras 81 generaciones, para el experimento C se detuvo tras 98 generaciones y para el experimento D tras 105 generaciones.



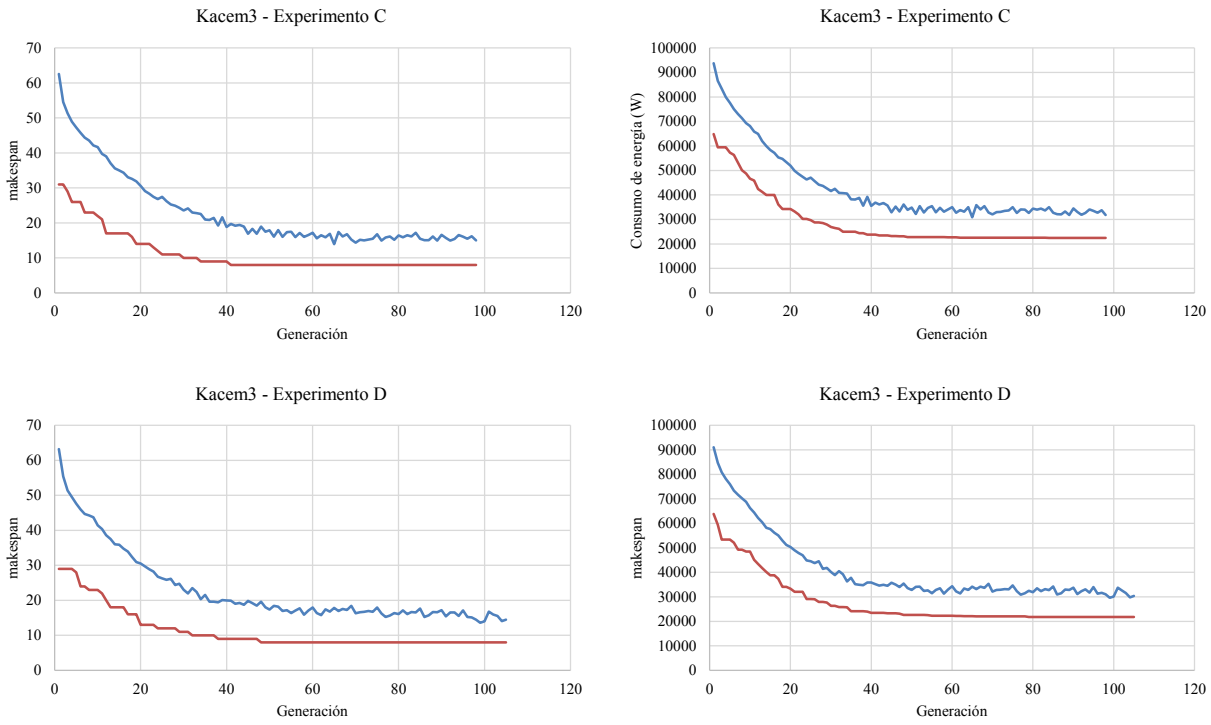


Figura 25. Desempeño del algoritmo genético propuesto. Instancia ka10x10.

5 Condusiones y trabajo futuro

La programación convencional de operaciones en entornos flexibles de producción job shop se enfoca en medidas de desempeño centralizadas y con foco en tiempos de ejecución. Sin embargo, en el contexto de la cuarta revolución industrial, la planificación debe adaptarse a sistemas de fabricación con objetivos adyacentes a la producción. Por lo tanto, es esencial que el enfoque de investigación en la programación de operaciones evolucione hacia modelos multiobjetivo que consideren otros factores relacionados con los recursos tecnológicos o máquinas, como el consumo de energía, la emisión de dióxido de carbono, la cantidad de mantenimientos por año y los recursos humanos, como la fatiga muscular o las afectaciones cognitivas. Los modelos de optimización pueden integrarse con los avances tecnológicos emergentes de la Industria 4.0. Estos modelos tienen la capacidad de definir programaciones en tiempo real considerando diferentes situaciones y evaluando diferentes medidas de desempeño.

El objetivo principal de esta investigación es desarrollar un modelo de programación de producción de un ambiente de manufactura Flexible Job Shop, en el que se logre la minimización y un balance simultaneo entre el *makespan* y el consumo energético de la producción. Este modelo guiará la ejecución de la programación de actividades y evaluará diferentes medidas de desempeño, particularmente, el tiempo de ejecución de la producción (*makespan*) y el consumo de energía. Para lograr este objetivo, se estableció un contexto de investigación, en el que se presentó la definición de los desafíos de la programación en entornos job shop, se caracterizaron los métodos tradicionales de optimización para abordar estos problemas y las medidas de desempeño asociadas, se caracterizaron los objetivos de consumo energético y se describió el enfoque propuesto en este trabajo para resolver el problema de programación de operaciones flexible. Desde este punto de partida, este trabajo se propuso hacer las siguientes contribuciones.

La primera contribución se centra en los resultados de la revisión de la literatura. Primero, se encontró que la programación de operaciones multiobjetivo es aún un tema en desarrollo y un enfoque que puede mejorar los resultados de la ejecución real de la programación de operaciones. Segundo, se encontró que en investigaciones previas con enfoque en consumo de energía se considera únicamente el tiempo de operación y no otros estados como la espera o el alistamiento, razón por la cual este trabajo considero dichos estados. Finalmente, la revisión de la literatura mostró que el cumplimiento de plazos predefinidos para los clientes ya no satisface por completo las demandas actuales y que resulta esencial demostrar que los procesos implementados tienen un impacto ambiental reducido.

La segunda contribución fue la definición y programación de una técnica metaheurística, algoritmo genético, para solucionar el problema de programación de operaciones: *flexible job shop scheduling problem*, y que incorpora y mide múltiples objetivos en entornos de manufactura. Los objetivos definidos en este trabajo fueron tiempo de ejecución total de la programación, *makespan* y consumo total de energía incorporando tres estados: estado en operación, estado en espera y estado

de alistamiento, cuya dirección de optimización fue minimización. La implementación y validación del algoritmo con las instancias de Kacem y Brandimarte con cuatro experimentos diseñados para evaluar su desempeño mostraron buenos resultados en la obtención de fronteras óptimas de Pareto, en la diversidad de las soluciones y por ende en el balance de los objetivos analizados y en la minimización de los objetivos.

El alcance del estudio fue limitado a la programación del algoritmo genético de ordenamiento no dominado (NSGA-II), como solución para el problema *Flexible Job Shop* considerando dos objetivos, eficiencia medida como *makespan* y efectividad, considerada como consumo energético. Además de evaluar el desempeño del modelo de solución propuesto a partir de comparaciones halladas en la literatura, que permitan evaluar la calidad de las respuestas obtenidas y los tiempos computacionales.

De este trabajo de investigación pueden derivarse diferentes trabajos futuros. Estos están relacionados con la implementación del modelo propuesto y el desarrollo del algoritmo propuesto. En relación con la implementación, se establecen las bases funcionales de la medición del consumo de energía. Se propone seguir cada recurso asociado en el piso para caracterizarlo de la mejor manera y asociar su consumo energético.

En relación con el desarrollo del algoritmo, se propone la integración de objetivos adicionales que brinden una caracterización más amplia del piso de manufactura. Esto es de particular importancia si se quiere tener la mayor abstracción del piso en el modelo. Adicionalmente, se propone la ejecución del modelo en entornos reales, ya sean en laboratorios de simulación o en plantas de manufactura, para validar las reducciones y los beneficios mostrados por el algoritmo a nivel teórico.

Bibliografía

Amjad, M. K., Butt, S. I., Kousar, R., Ahmad, R., Agha, M. H., Faping, Z., Anjum, N., & Asgher, U. (2018). Recent research trends in genetic algorithm based flexible job shop scheduling problems. *Mathematical Problems in Engineering*, *2018*, 1–32. https://doi.org/10.1155/2018/9270802

Brandimarte, P. (1993). Routing and scheduling in a flexible job shop by tabu search. *Annals of Operations Research*, *41*(3), 157–183. https://doi.org/10.1007/BF02023073

Chan, F. T. S., Wong, T. C., & Chan, L. Y. (2006). Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, *44*(11), 2071–2089. https://doi.org/10.1080/00207540500386012

Da Silveira, G., Borenstein, D., Fogliatto, F. S., Da Silveira, G., Borenstein, D., & Fogliatto, F. S. (2001). Mass customization: Literature review and research directions. *International Journal of Production Economics*, *72*(1), 1–13.

Dai, M., Tang, D., Giret, A., & Salido, M. A. (2019). Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. *Robotics and Computer-Integrated Manufacturing*, *59*, 143–157. https://doi.org/10.1016/j.rcim.2019.04.006

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*. https://direct.mit.edu/evco/article-abstract/7/3/205/855

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017

Del Val Román, J. L. (2018). Industria 4.0: la transformación digital de la industria. *Informe de La Conferencia de Directores y Decanos de Ingeniería Informática CODDII*.

Garey, M. R., & Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. In *W.H. Freeman*. W.H. Freeman. https://doi.org/10.1137/1024022

Guo, C., & Lei, D. (2018a). A two-phase variable neighborhood search for flexible job shop scheduling problem with energy consumption constraint. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10954 LNCS*, 708–716. https://doi.org/10.1007/978-3-319-95930-6_72

Guo, C., & Lei, D. (2018b). Multi-objective flexible job shop scheduling problem with energy consumption constraint using imperialist competitive algorithm. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10954 LNCS*, 659–669. https://doi.org/10.1007/978-3-319-95930-6_66

Hemmati Far, M., Haleh, H., & Saghaei, A. (2019). A fuzzy bi-objective flexible cell scheduling optimization model under green and energy-efficient strategy using Pareto-based algorithms: SATPSPGA, SANRGA, and NSGA-II. *International Journal of Advanced Manufacturing Technology*, *105*(9), 3853–3879. https://doi.org/10.1007/S00170-019-03797-W

Jiang, E., & Wang, L. (2019). A modified MOEA/D for energy-efficient Flexible Job Shop Scheduling Problem. *IEEE International Conference on Industrial Engineering and Engineering Management*, *2019-December*, 1476–1480. https://doi.org/10.1109/IEEM.2018.8607582

Jiang, T., & Deng, G. (2018). Optimizing the Low-Carbon Flexible Job Shop Scheduling Problem Considering Energy Consumption. *IEEE Access*, *6*, 46346–46355. https://doi.org/10.1109/ACCESS.2018.2866133

Jiang, Z., Zuo, L., & Mingcheng, E. (2014). Study on multi-objective flexible job-shop scheduling problem considering energy consumption. *Journal of Industrial Engineering and Management*, *7*(3), 589–604. https://doi.org/10.3926/jiem.1075

Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, *1*(1), 61–68. https://doi.org/10.1002/nav.3800010110

Jones, D. F., Mirrazavi, S. K., & Tamiz, M. (2002). Multi-objective meta-heuristics: An overview of the current state-of-the-art. *European Journal of Operational Research*, *137*(1), 1–9. https://doi.org/10.1016/S0377-2217(01)00123-0

Kacem, I., Hammadi, S., & Borne, P. (2002). Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic. *Mathematics and Computers in Simulation*, *60*(3–5), 245–276. https://doi.org/10.1016/S0378-4754(02)00019-8

Kaoutar, S., & Mohamed, E. (2017). Multi-criteria optimization of neural networks using multi-objective genetic algorithm. *Intelligent Systems and Computer Vision*. https://doi.org/10.1109/ISACV.2017.8054962

Lei, D., Li, M., & Wang, L. (2019). A Two-Phase Meta-Heuristic for Multiobjective Flexible Job Shop Scheduling Problem with Total Energy Consumption Threshold. *IEEE Transactions on Cybernetics*, *49*(3), 1097–1109. https://doi.org/10.1109/TCYB.2018.2796119

Lei, D., Zheng, Y., & Guo, X. (2017). A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption. *International Journal of Production Research*, *55*(11), 3126–3140. https://doi.org/10.1080/00207543.2016.1262082

Li, Y., He, Y., Wang, Y., Tao, F., & Sutherland, J. W. (2020). An optimization method for energy-conscious production in flexible machining job shops with dynamic job arrivals and machine breakdowns. *Journal of Cleaner Production*, *254*. https://doi.org/10.1016/J.JCLEPRO.2020.120009

Liang, X., Chen, J., Gu, X., & Huang, M. (2021). Improved Adaptive Non-Dominated Sorting Genetic Algorithm with Elite Strategy for Solving Multi-Objective Flexible Job-Shop Scheduling Problem. *IEEE Access*, *9*, 106352–106362. https://doi.org/10.1109/ACCESS.2021.3098823

Liao, C. J., Gen, M., Tiwari, M. K., & Chang, P. C. (2013). Meta-heuristics for manufacturing scheduling and logistics problems. *International Journal of Production Economics*, *141*(1), 1–3. https://doi.org/10.1016/J.IJPE.2012.09.004

Liao, W., & Wang, T. (2018). Promoting green and sustainability: A multi- objective optimization method for the job-shop scheduling problem. *Sustainability (Switzerland)*, *10*(11). https://doi.org/10.3390/su10114205

Liu, Q., Gui, Z., Xiong, S., & Zhan, M. (2021). A principal component analysis dominance mechanism based many-objective scheduling optimization. *Applied Soft Computing*, *113*, 107931. https://doi.org/10.1016/J.ASOC.2021.107931

López Ramón y Cajal, J., & Escudero Ceballos, V. (2016). Industria 4.0, la gran oportunidad. *Economía Aragonesa, ISSN 1576-7736, Nº. 59, 2016, Págs. 109-123*, *59*, 109–123.

Luke, S. (2013). *Essentials of Metaheuristics* (2nd. ed.). Lulu. https://cs.gmu.edu/~sean/book/metaheuristics/

Manne, A. S. (1960). On the job-shop scheduling problem. *Operations Research*, *8*(2), 219–223. https://doi.org/10.1287/opre.8.2.219

Marzouki, B., Belkahla Driss, O., & Ghédira, K. (2017). Multi Agent model based on Chemical Reaction Optimization with Greedy algorithm for Flexible Job shop Scheduling Problem. *Procedia Computer Science*, *112*, 81–90. https://doi.org/10.1016/J.PROCS.2017.08.174

Mastrolilli, M., & Gambardella, L. M. (2000). Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling*.

Meng, L., Zhang, C., Zhang, B., & Ren, Y. (2019). Mathematical Modeling and Optimization of Energy-Conscious Flexible Job Shop Scheduling Problem with Worker Flexibility. *IEEE Access*, *7*, 68043–68059. https://doi.org/10.1109/ACCESS.2019.2916468

Mokhtari, H., & Hasani, A. (2017). An energy-efficient multi-objective optimization for flexible job-shop scheduling problem. *Computers and Chemical Engineering*, *104*, 339–352. https://doi.org/10.1016/j.compchemeng.2017.05.004

Pach, C., Berger, T., Sallez, Y., & Trentesaux, D. (2015). Reactive control of overall power consumption in flexible manufacturing systems scheduling: A Potential Fields model. *Control Engineering Practice*, *44*, 193–208. https://doi.org/10.1016/J.CONENGPRAC.2015.08.003

Piroozfard, H., Wong, K. Y., & Wong, W. P. (2018). Minimizing total carbon footprint and total late work criterion in flexible job shop scheduling by using an improved multi-objective genetic algorithm. *Resources, Conservation and Recycling*, *128*, 267–283. https://doi.org/10.1016/j.resconrec.2016.12.001

Rahmati, S. H. A., Zandieh, M., & Yazdani, M. (2013). Developing two multi-objective evolutionary algorithms for the multi-objective flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *64*(5–8), 915–932. https://doi.org/10.1007/s00170-012-4051-1

República de Colombia, Ministerio de Minas y Energía, & Unidad de Planeación Minero-Energética. (2016). *Plan de Acción Indicativo de Eficiencia Energética 2017-2022*.

Saad, I., Boukef, H., & Borne, P. (2007). The comparison of criteria aggregative approaches for the multiobjective optimization of flexible job-shop scheduling problems. *IFAC Proceedings Volumes*, *40*(18), 603–609. https://doi.org/10.3182/20070927-4-RO-3905.00100

Salido, M. A., Escamilla, J., Barber, F., & Giret, A. (2017). Rescheduling in job-shop problems for sustainable manufacturing systems. *Journal of Cleaner Production*, *162*, S121–S132. https://doi.org/10.1016/j.jclepro.2016.11.002

Seng, D. W., Li, J. W., Fang, X. J., Zhang, X. F., & Chen, J. (2018). Low-carbon flexible job-shop scheduling based on improved nondominated sorting genetic algorithm-II. *International Journal of Simulation Modelling*, *17*(4), 712–723. https://doi.org/10.2507/IJSIMM17(4)CO18

Singh, M. R., Singh, M., Mahapatra, S. S., & Jagadev, N. (2016). Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem. *International Journal of Advanced Manufacturing Technology*, *85*(9–12), 2353–2366. https://doi.org/10.1007/s00170-015-8075-1

Sule, D. R. (2021). *Production Planning and Industrial Scheduling: examples, case studies and applications*. ROUTLEDGE.

Talbi, E.-G. (2009). *Metaheuristics: From Desing to Implementation*. 624.

Wagner, H. M. (1959). An integer linear programming model for machine scheduling. *Naval Research Logistics Quarterly*, *6*(2), 131–140. https://doi.org/10.1002/nav.3800060205

Wang, H., Jiang, Z., Wang, Y., Zhang, H., & Wang, Y. (2018). A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization. *Journal of Cleaner Production*, *188*, 575–588. https://doi.org/10.1016/j.jclepro.2018.03.254

Wang, J., Yang, J., Zhang, Y., Ren, S., & Liu, Y. (2020). Infinitely repeated game based real-time scheduling for low-carbon flexible job shop considering multi-time periods. *Journal of Cleaner Production*, *247*, 119093. https://doi.org/10.1016/J.JCLEPRO.2019.119093

Wu, X., & Sun, Y. (2018). A green scheduling algorithm for flexible job shop with energy-saving measures. *Journal of Cleaner Production*, *172*, 3249–3264. https://doi.org/10.1016/j.jclepro.2017.10.342

Xia, W., & Wu, Z. (2005). An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. *Computers & Industrial Engineering*, *48*(2), 409–425. https://doi.org/10.1016/J.CIE.2005.01.018

Xu, W., Hu, Y., Luo, W., Wang, L., & Wu, R. (2021). A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission. *Computers & Industrial Engineering*, *157*, 107318. https://doi.org/10.1016/J.CIE.2021.107318

Yin, L., Li, X., Gao, L., Lu, C., & Zhang, Z. (2017). A novel mathematical model and multi-objective method for the low-carbon flexible job shop scheduling problem. *Sustainable Computing: Informatics and Systems*, *13*, 15–30. https://doi.org/10.1016/j.suscom.2016.11.002

Zhang, J., Ding, G., Zou, Y., Qin, S., & Fu, J. (2019). Review of job shop scheduling research and its new perspectives under Industry 4.0. *Journal of Intelligent Manufacturing*, *30*(4), 1809–1830. https://doi.org/10.1007/s10845-017-1350-2

Zhang, L., Tang, Q., Wu, Z., & Wang, F. (2017). Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops. *Energy*, *138*, 210–227. https://doi.org/10.1016/j.energy.2017.07.005

Zhang, R., & Chiong, R. (2016). Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *Journal of Cleaner Production*, *112*, 3361–3375. https://doi.org/10.1016/j.jclepro.2015.09.097

Zhang, X., Ji, Z., & Wang, Y. (2018). An improved SFLA for flexible job shop scheduling problem considering energy consumption. *Modern Physics Letters B*, *32*(34–36). https://doi.org/10.1142/S0217984918401127

Zhou, G., Chen, Z., Zhang, C., & Chang, F. (2022). An adaptive ensemble deep forest based dynamic scheduling strategy for low carbon flexible job shop under recessive disturbance. *Journal of Cleaner Production*, *337*, 130541. https://doi.org/10.1016/J.JCLEPRO.2022.130541

Apéndice A: parámetros del algoritmo

Este apéndice presenta el diseño de experimento realizado sobre los parámetros del algoritmo genético para estudiar el efecto de estos sobre la variable de salida, *makespan*. Se realizó un diseño de experimentos factorial 3^k con $k = 3$ (tres factores).

Se definieron los siguientes factores:

(1) Puntos de entrecruzamiento (*c*)
(2) Probabilidad de mutación (*mr*)
(3) Umbral de aceptación (*thv*)

Para cada factor se definieron tres niveles, denominados como bajo, medio y alto. La Tabla 15 presenta los valores cuantitativos de los niveles para cada factor.

Tabla 15. Arreglo factorial.

Factor	Niveles		
	Bajo	Medio	Alto
Puntos de entrecruzamiento	2	5	10
Probabilidad de mutación	0,25	0,50	0,75
Umbral de aceptación	0,01	0,05	0,10

A partir de la información de la tabla anterior y considerando todas las posibilidades de combinación de los niveles de los factores, se forman 27 tratamientos. Para cada uno de los tratamientos se llevan a cabo diez (10) replicas.

En el presente diseño de experimentos, los tratamientos se realización usando la instancia 'Mk01' propuesta por Brandimarte (1993).

La Tabla 16 muestra el análisis de varianza (ANOVA) realizado tras la realización completa del experimento. Este análisis permite saber si los efectos principales y de interacción son estadísticamente significativos. Con el fin de probar las hipótesis de los efectos activos, se define un valor de significancia $\alpha = 0,05$.

Tabla 16. Análisis de varianza.

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Model	26	14128,1	543,39	323,82	0,000
Linear	6	13608,3	2268,04	1351,58	0,000
c_points	2	42,8	21,40	12,75	0,000
threshold	2	9552,7	4776,36	2846,34	0,000
mut_r	2	4012,7	2006,37	1195,64	0,000
2-Way Interactions	12	464,5	38,71	23,07	0,000
c_points*threshold	4	46,6	11,65	6,94	0,000
c_points*mut_r	4	13,7	3,41	2,03	0,088
threshold*mut_r	4	404,2	101,06	60,22	0,000
3-Way Interactions	8	55,3	6,91	4,12	0,000
c_points*threshold*mut_r	8	55,3	6,91	4,12	0,000
Error	513	860,9	1,68		
Total	539	14988,9			

Del ANOVA se concluye que los tres efectos individuales, c_points: puntos de entrecruzamiento (c), threshold: umbral de aceptación (thv) y mut_r: probabilidad de mutación (mr) están activos. Con respecto a los efectos de interacción, el único efecto no activo es el definido como c_points*mut_r. La significancia de la interacción detectada por el ANOVA se observa en el hecho de que las líneas de la Figura 26 y Figura 27 tienen pendientes diferentes.

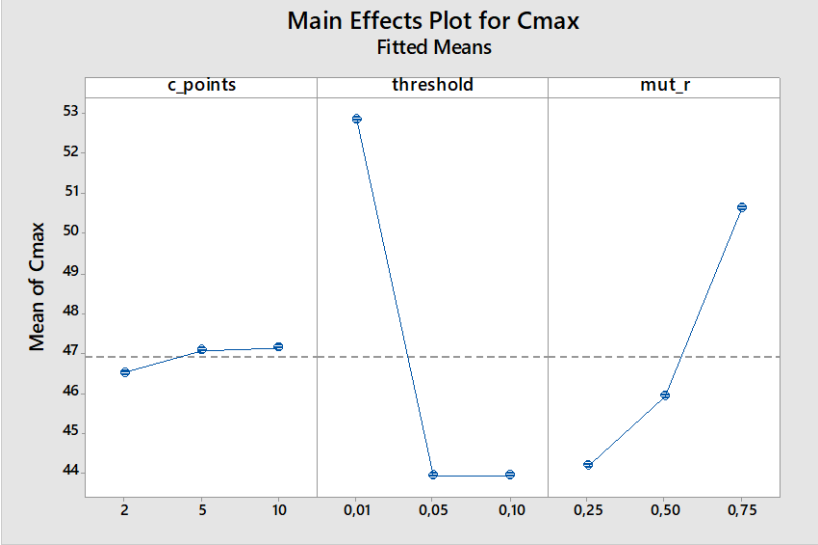


Figura 26. Efectos principales del experimento

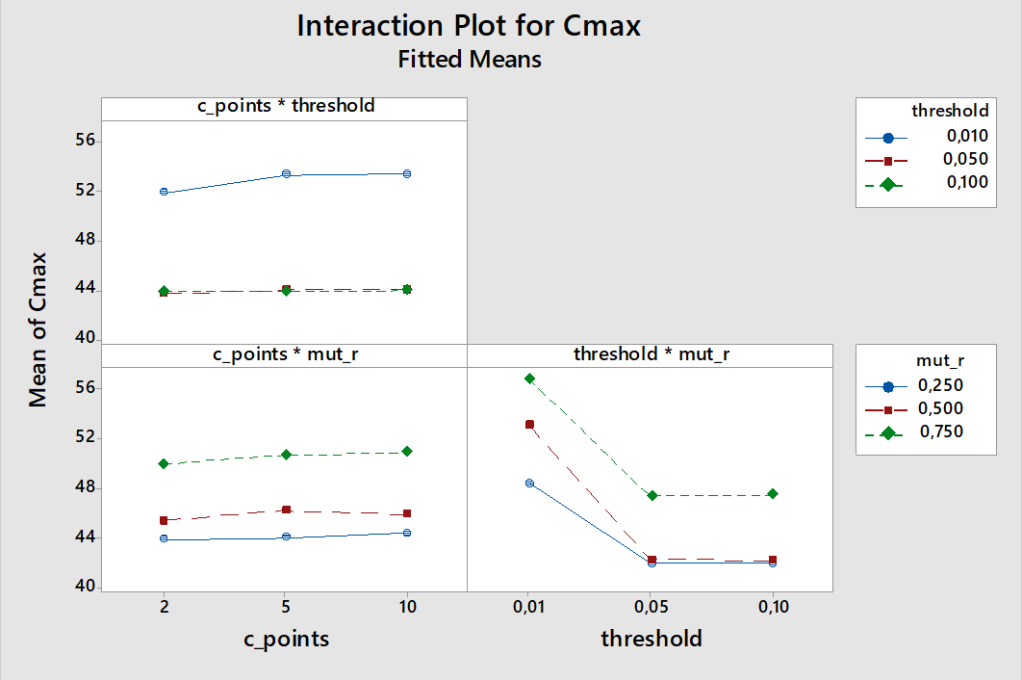


Figura 27. Efectos de interacción del experimento.

Por lo tanto, las condiciones de operación (parámetros del algoritmo) que convienen son:

(1) Puntos de entrecruzamiento (c) = 2

(2) Probabilidad de mutación (mr) = 0,25

(3) Umbral de aceptación (thv) = 0,10

Apéndice B: código de programación en Python.

Este apéndice presenta el código en Python de las operaciones de ordenamiento no dominado del algoritmo genético.

Ordenamiento no dominado

```
def fast_sorting(P):
    '''
    Función.
    Encuentra la frontera de la población analizada. La función codificada está basada en el seudocódigo indicado en Sección 3.3.3
    del presente documento.

    Parámetros:
    - P: población a organizar
    '''

 Pprima = [P[0]]                          # includes first member in P'
 for p in range(1, len(P)):                # take one solution at a time
    p_temp = P[p]
    Pprima.append(p_temp)                      # include p in P' temporarily
    Pprima1 = Pprima.copy()
    for q in Pprima1:                        # compare p with other members of P'
        if q != p_temp:
            if dominance(q, p_temp) == True:    # if p dominates mebers of P' (q),
                Pprima.remove(q)               # delete it
            if dominance(p_temp, q) == True:    # if p is dominated by other mebers of P',
                Pprima.remove(p_temp)          # do not include in P'
                break

 return Pprima
```

Función de dominancia

```
def dominance(A, B):
  '''
    Función.
    Evalúa la dominancia de un cromosoma sobre otro a partir de los valores de las funciones objetivo. La dominancia
    evaluada es de minimización.

    Parámetros:
      - A --> cromosoma a evaluar 1
      - B --> cromosoma a evaluar 2

  La función extrae los valores [0] y [1] que corresponden a los valores de f1 y f2.
  '''
```

```
        x1 = A[0]
        y1 = A[1]
        x2 = B[0]
        y2 = B[1]

        ## It proofs if B dominates A
        if x2 <= x1 and y2 <= y1 and (x2 < x1 or y2 < y1):
            result = True
        else:
            result = False

        return result
```

Creación de fronteras óptimas de Pareto

```
def create_fronts (P):
    '''
    Encuentra las fronteras de cada uno de los miembros de la población analizada.

    Parámetros:
        - P: población a organizar
    '''
    fronts = []
    P_new = P

    while len(P_new) > 0:
        F = fast_sorting(P_new)
        fronts.append(F)
        P_new = [i for i in P_new if i not in F]

    return fronts
```

Distancia de aglomeración

```
def crowding_distance(fronts):
    '''
    Función.
    Encuentra la "densidad" de cada punto (crowding-distance). La función codificada esta basada en el seudocódigo presentado
en la sección 3.3.3 del presente documento.

    Parámetros:
        - fronts --> puntos de cada frontera a calcular.

    Tiene como retorno la información de la distancia en el indice [3] de la población.
    '''
    for f in range(len(fronts)):
```

```
    l = len(fronts[f])
    for obj in range(2):
        fronts[f].sort(key = lambda item: item[obj])
        fronts[f][0][2] = fronts[f][l - 1][2] = float('inf')
        for i in range(1, l - 1):
            fronts[f][i][2] = fronts[f][i][2] + fronts[f][i + 1][obj] - fronts[f][i - 1][obj]

return fronts
```