UNIVERSIDAD PANAMERICANA

CAMPUS GUADALAJARA

POSGRADOS DE INGENIERÍA

Métodos de solución para problemas de secuenciación en máquinas paralelas no relacionadas

Tesis realizada por:

Héctor Roberto García De Alba Valenzuela

Director:

Dr. Samuel Moises Nucamendi Guillen

Codirector:

Dr. Oliver Avalos Rosales

Tesis presentada para optar por el grado de Doctor en Ingeniería con Reconocimiento de Validez Oficial de Estudios de la SECRETARIA DE EDUCACIÓN PÚBLICA, según acuerdo 20170049 con fecha 18-IV-2017.

Zapopan, Jalisco, Mayo 2023

Doctorado en Ingeniería
Posgrados de Ingeniería

Sinodales:

Dr. Oliver Avalos Rosales
Profesor Investigador
Centro de Investigación en Matemáticas Aplicadas
Universidad Autónoma de Coahuila

Dr. Samuel Moisés Nucamendi Guillén
Profesor Investigador Titular D
Facultad de Ingeniería
Universidad Panamericana

Dr. Elías Olivares Benítez
Profesor Investigador Titular D
Facultad de Ingeniería
Universidad Panamericana

Dr. Adrián Ramírez Nafarrate
Director Regional del Departamento de Ingeniería Industrial
Escuela de Ingeniería y Ciencias
Tecnológico de Monterrey

Dra. Avelina Alejo Reyes
Profesora Investigadora
Facultad de Ingeniería
Universidad Panamericana

Resumen:

En los entornos productivos se presentan constantemente situaciones en las que se tiene un conjunto de actividades por realizarse, un conjunto de máquinas capaces de realizar dichas tareas, y se debe decidir cómo se debe de repartir la carga de trabajo y en qué orden se deben de realizar dichas actividades. Estos sistemas se estudian en las disciplinas de optimización combinatoria bajo los conceptos de secuenciación, y son relevantes en la industria debido a las grandes implicaciones económicas asociadas al uso de recursos que dichas decisiones impactan en diversas aplicaciones, como lo puede ser las industrias de impresión, manufactura de herramentales o artículos personalizados y de alta precisión, talleres de manufactura flexible, entre otras.

En este trabajo de investigación se estudian dos tipos de problemas de secuenciación. El primero consiste en un problema de secuenciación en máquinas paralelas no relacionadas con minimización de la tardanza total, mientras que el segundo se encuentra relacionado con la secuenciación en máquinas paralelas no relacionadas con minimización de makespan, tiempos de preparación dependientes de la secuencia y recurso compartido.

Para el primer problema se presentan 2 metodologías de solución: la primera utiliza un modelo de programación lineal entera mixta basado en variables de posición, el cual no necesita de restricciones de gran *M* e incorpora desigualdades válidas para acelerar su ejecución, y como segunda metodología se presenta un algoritmo metaheurístico de búsqueda local iterada con estrategias de mejora basadas en mecanismos de reinserción e intercambio de tareas.

El segundo problema bajo estudio considera una extensión del problema de recursos compartidos, donde dicho recurso ejecuta todos los procesos de preparación y debe de tener una secuencia propia de actividades. Esta secuencia, a su vez, está ligada a las secuencias de las distintas máquinas del sistema, visualizando así al recurso compartido como una máquina. Para este problema se

exploran 2 metodologías de solución; primero un conjunto de modelos (6 específicamente) de programación lineal entera mixta que describen el sistema desde varias perspectivas relacionadas a hitos de tiempo del sistema. La segunda metodología de solución contempla un par de algoritmos metaheurísticos de descenso por vecindarios variables (4 mecanismos de generación de vecindarios), e incorpora dos mecanismos de diversificación.

Para el primer problema se tuvo la oportunidad de comparar el desempeño del modelo matemático contra un modelo previamente presentado en la literatura, demostrando su superioridad al obtener soluciones óptimas en instancias donde el modelo previo reportaba soluciones factibles con hasta 74% de gap. Adicionalmente, se mejoró el desempeño del modelo propuesto mediante la inclusión de desigualdades válidas, resultando en reducciones de hasta un 30% en tiempo de procesamiento.

Para el segundo problema se consideraron 6 formulaciones distintas, basadas en diferentes momentos de referencia relevantes para el sistema, con lo que se determinó que las formulaciones que usan como referencia el tiempo de terminación del proceso de preparación tienen mejor desempeño.

Para ambos problemas se realizaron comparativas entre los modelos y los algoritmos propuestos correspondientes, con el fin de determinar la capacidad de los algoritmos para encontrar soluciones de alta calidad. Los resultados experimentales muestran que los algoritmos propuestos son competitivos. Además, para el primer problema se lograron abordar instancias de hasta 400 tareas mediante dicho algoritmo, mientras que, para el segundo, se resuelven instancias de hasta 60 tareas.

Índice de contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

1. Introducción.

La competitividad en el mercado actual demanda a los productores de bienes y servicios a revisar y mejorar sus procesos. Industrias ya establecidas como la automotriz, buscan el orden específico para ensamblar vehículos buscando aumentar la producción; industrias dedicadas al transporte de bienes buscan las rutas que reduzcan sus tiempos de traslado; industrias dedicadas a prestar servicios especializados, buscan la manera de usar mejor sus activos para reducir sus costos, y muchos ejemplos más pueden ser nombrados de sectores en donde, aunque existen soluciones en la actualidad, se siguen buscando nuevas y mejores alternativas para que se aumenten los beneficios obtenidos, se reduzcan los recursos requeridos y que, en general, ofrezcan una ventaja competitiva.

Debido a esto, la búsqueda de mejores soluciones para los distintos problemas de la industria ha derivado en la necesidad de una disciplina dedicada a obtener dichas respuestas. La investigación de operaciones (IO) se dedica a estudiar esta clase de problemas, en los cuales existe una serie de posibles soluciones que son calificadas mediante una función de valor. Sin embargo, la exploración de dichas soluciones de manera eficiente sigue siendo un reto para la esta disciplina. Una alternativa es implementar técnicas como la programación lineal, que pretende establecer adecuadamente un espacio de soluciones, el cual es explorado mediante un método exacto para así poder encontrar la respuesta óptima. Desafortunadamente, este enfoque no siempre es viable para problemas grandes o de alta complejidad, los cuales suelen aparecer con gran frecuencia en aplicaciones de la industria.

Existe una fuerte demanda de nuevas técnicas de optimización, y refinamiento de las ya existentes que puedan mantenerse a la par de los nuevos retos presentados en la industria actual. Uno de los retos importantes que se estudian en la investigación de operaciones es el problema de secuenciación, en el que para una serie de tareas debe decidirse cuándo, cómo y dónde deben ser

procesadas. En esta tesis se discutirán dos variantes específicas para dicha clase de problemas y se propondrán nuevas técnicas (exactas y aproximadas) para su resolución.

1.1. Descripción del problema.

Este trabajo de tesis aborda el estudio de problemas de secuenciación de tareas en máquinas paralelas no relacionadas. Bajo este enfoque, cualquier trabajo puede ser procesado en cualquier máquina, pero el tiempo necesario para procesar el trabajo dependerá según la máquina asignada. Es posible asignar cualquier cantidad de trabajos por máquina, y cada máquina sólo puede procesar un trabajo a la vez. Una vez iniciado el procesamiento de un trabajo no puede ser interrumpido. El problema pretende determinar el orden o secuencia en el que cada máquina procesará todos sus trabajos asignados, buscando, como objetivo, optimizar métricas específicas de desempeño.

En particular se estudiarán dos problemas; el primero, en el que se considerará una función objetivo que cuantifica la tardanza total del sistema, y el segundo (y más complejo), en el que se considerarán tiempos de preparación dependientes de la secuencia, así como un recurso compartido que realizará dichas preparaciones y en el cual se tendrá como función objetivo la minimización del *makespan (*entendido éste como el momento en el que todas las tareas han sido completadas).

En cuanto al primer problema, cada tarea contará con una fecha de entrega, la cual puede o no ser cumplida. Terminar una tarea antes no tiene ninguna repercusión para el sistema; sin embargo, terminar una tarea después de su fecha de entrega se considera indeseable. A dicho evento se le llamará tardanza de la tarea y el propósito de este problema es el intentar minimizar la suma de las tardanzas asociadas a cada una de las tareas. Este problema será referido en este

trabajo como *“Problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza”.* Un ejemplo de este sistema se observa en las industrias de prototipado mediante impresión 3D, que, al ser una tecnología relativamente nueva, constantemente se presentan en el mercado nuevos equipos de impresión con características nuevas. Sin embargo, los equipos anteriores siguen siendo totalmente funcionales, resultando en una capacidad de producción instalada con características mixtas en los distintos equipos.

En cuanto al segundo problema, los tiempos para realizar la preparación pueden variar, en primera instancia, según la máquina a la que se asigne el trabajo, ya que los trabajos pueden necesitar una configuración distinta en las distintas máquinas. El tiempo de preparación también variará de acuerdo con la secuencia, es decir, del trabajo previamente realizado en la máquina donde es procesada, ya que un par de tareas secuenciadas podrían ser tan similares que la tarea posterior necesitará pocos cambios en la configuración de la máquina para ser procesada, o bien ser tan diferente que se necesitarían cambios más grandes en la configuración de la máquina.

Para el segundo problema, los procedimientos de preparación serán realizados por un único operador (el cual será llamado recurso compartido), que irá desplazándose entre máquinas realizando la preparación de cada trabajo a ser procesado a continuación, según las secuencias asociadas a cada máquina. Dicho proceso tampoco puede interrumpirse una vez iniciado, ni efectuarse simultáneamente. Esto resulta en restricciones de traslape en las preparaciones. Por lo tanto, al operador que realiza las preparaciones también se le asignará una secuencia de procesamiento, determinando el orden en el que se realizarán los procesos de preparación. Esta secuencia estará fuertemente relacionada a las secuencias de procesamiento de las máquinas. Una vez finalizada la preparación de una tarea, la máquina procesará el trabajo inmediata e independientemente (sin necesitar del operador). Para este enfoque, se tiene el objetivo de minimizar el Makespan. Este problema se definirá como “*Problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de*

secuencia, recurso compartido, y minimización del makespan". De manera similar al problema anterior, se puede visualizar este sistema en las industrias de prototipado, donde consideraríamos a un empleado que monta herramentales y materiales de impresión en los equipos, dependiendo de el tipo de tarea que cada maquina va a realizar durante su secuencia, y dichos procesos de preparación podrían requerir menos tiempo en secuencias de tareas que utilizan los mismos herramentales, o mas tiempo en secuencias donde las tareas requieran el cambio de herramental.

1.2. Motivación.

Los sistemas de procesamiento en máquinas paralelas se pueden encontrar con gran variedad en la industria, desde las más modernas, como lo es el diseño y prototipado de herramientas, hasta las más antiguas como lo es la orfebrería. En la industria del prototipado de herramientas podemos encontrar equipos de impresión 3D que trabajan en paralelo para crear las distintas piezas especializadas, cuyos diseños requieren de alta precisión. Análogamente encontramos en la industria de la orfebrería un taller con artesanos, los cuales atienden órdenes para la creación de piezas de joyería personalizada. Ambas industrias necesitan establecer estrategias y tomar decisiones con respecto al uso de sus recursos en estos sistemas; cómo asignar las actividades, en qué momento realizarlas, qué recursos utilizar, etc. Debido a lo anterior se hace evidente la necesidad de herramientas que faciliten la toma de dichas decisiones de manera óptima.

Se han realizado esfuerzos en la disciplina de investigación de operaciones, matemáticas, ciencias de la computación, entre otras, para diseñar las herramientas que permitan abordar y resolver los problemas de secuenciación en máquinas paralelas. Sin embargo, existen aún problemas cuyas propuestas no han sido capaces de mantenerse a la altura de las necesidades de la industria, y existen otros

problemas que todavía no han sido estudiados. La motivación de esta tesis es aportar al estado del arte en estas áreas de investigación, proponiendo nuevas teorías para problemas ya conocidos, y abordando aquellos que aún quedan sin tratar o no han sido identificados.

1.3. Objetivos de investigación

Los objetivos de investigación se especifican según su alcance, clasificándose en generales y específicos, que se detallan a continuación:

1.3.1. Objetivo general.

La presente tesis tiene como objetivo general, formular modelos de programación lineal entera mixta que sean capaces de describir y resolver los problemas bajo estudio, así como desarrollar algoritmos metaheurísticos que permitan obtener soluciones de alta calidad en un tiempo computacional razonable, siendo dichos algoritmos representativos de las características específicas de cada problema.

1.3.2. Objetivos específicos.

Adicionalmente, se contará con los siguientes objetivos específicos:

- Para cada problema bajo estudio, presentar una formulación matemática basada en programación lineal que sea capaz de describirlo adecuadamente.
- Para cada modelo propuesto, realizar experimentación para encontrar los límites de tamaños de instancias que puede abordar.

- Para cada problema bajo estudio, diseñar un algoritmo metaheurístico capaz de abordar instancias de mayor tamaño que el límite encontrado para el modelo lineal.

- Para cada algoritmo propuesto, realizar experimentación para determinar su desempeño en comparación con el modelo lineal, en cuanto al tiempo de ejecución y calidad de la respuesta.

Debido a que el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza ya ha sido estudiado en la literatura, se tendrán los siguientes objetivos específicos exclusivos:

- Realizar experimentación comparativa del modelo propuesto, con otros modelos existentes en la literatura.

- Realizar experimentación comparativa para determinar las maneras más adecuadas de crear instancias.

Debido a que el problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido y minimización del Makespan, aún no ha sido estudiado en la literatura, se establecerán los siguientes objetivos específicos:

- Desarrollar diferentes versiones del modelo lineal basadas en distintos enfoques.

- Realizar experimentación para comparar las distintas versiones del modelo.

1.4. Justificación.

La justificación para abordar el problema de máquinas paralelas no relacionadas es sintetizada mediante los siguientes puntos:

- Problemas de esta índole se presentan constantemente en la industria, de manera muy diversa.

- El estudio de dichos problemas es relevante para disciplinas académicas, como ciencias de la computación y matemáticas aplicadas, donde puede aplicarse a sistemas de asignación de recursos computacionales para realizar cálculos en paralelo.

- Para el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza, es un sistema que ha sido poco estudiado en la literatura, y su estudio propone mejoras en la comprensión, diseño y aplicación de metodologías actuales, así como la proposición de nuevas.

- Para el problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido, y minimización del makespan, aún no ha sido estudiado en la literatura, por lo que se proponen estrategias nuevas que fungen como punto de partida para explorar nuevas áreas de investigación.

Las razones por las que se realizan las técnicas propuestas son las siguientes:

- En cuanto a la formulación de modelos de programación lineal entera; es una técnica conocida y probada que es capaz de obtener soluciones óptimas, las cuales servirán como punto de comparación para otros métodos a desarrollar.

- En cuanto al desarrollo de algoritmos metaheurísticos: son técnicas de fácil implementación, debido a que no requiere de conocimientos técnicos

sobre temas de modelación matemática, y que permite abordar instancias de mayor tamaño en tiempos computacionales menores que un modelo lineal entero, superando así limitantes técnicas que impiden la implementación de un modelo matemático para cualquier tamaño de instancia del problema.

1.5. Hipótesis.

Las hipótesis que se plantean para el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza son las siguientes:

- El modelo lineal propuesto será capaz de resolver a optimalidad, instancias de mayor tamaño que los modelos presentados en la literatura actual.
- Se pueden implementar desigualdades válidas que permiten reducir el tiempo computacional para la resolución del modelo.
- Es posible establecer un algoritmo que pueda alcanzar valores óptimos globales para instancias de hasta 150 tareas y 20 máquinas.
- El algoritmo metaheurístico será más veloz en tiempo de ejecución que el modelo lineal, sin sacrificar la calidad de la solución.
- El algoritmo metaheurístico es capaz de abordar instancias de mayor tamaño que el modelo lineal.
- Las metodologías de caracterización y creación de instancias en la literatura actual son adecuadas.

Las hipótesis que se plantean para el problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido, y minimización del Makespan, son las siguientes:

- Es posible diseñar modelos de programación lineal entera mixta que adecuadamente describan las características del problema.
- Los modelos de programación lineal entera mixta propuestos, son capaces de obtener óptimos globales para instancias de hasta 14 tareas y 4 máquinas.
- Es posible establecer un algoritmo que pueda alcanzar valores óptimos globales para instancias de hasta 14 tareas y 4 máquinas.
- El algoritmo metaheurístico podrá encontrar soluciones factibles de alta calidad, requiriendo tiempo computacional menor que los modelos lineales propuestos.
- El algoritmo metaheurístico es capaz de abordar instancias de tamaños que los modelos propuestos no son capaces de resolver.

1.6. Alcance.

En esta tesis se desarrollarán y probarán técnicas para resolver el problema de máquinas paralelas no relacionadas, con las dos variantes propuestas. Los procedimientos y estrategias propuestas seguirán las siguientes delimitaciones (para ambos problemas):

- Se considerará que todos los parámetros son fijos, conocidos y determinísticos.

- Se excluirá cualquier análisis estocástico.

En específico, para el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza, se tendrá:

- El modelo matemático propuesto será probado con instancias de hasta 150 tareas y 20 máquinas.
- El algoritmo metaheurístico será probado con instancias de hasta 400 tareas y 20 máquinas.

Y, para el problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido, y minimización del Makespan, se tendrá:

- Los modelos matemáticos propuestos serán probados con instancias de hasta 14 tareas y 4 máquinas.
- El algoritmo propuesto será probado con instancias de hasta 60 tareas y 8 máquinas.

1.7. Estructura de la tesis.

Este documento contará con los siguientes capítulos: El primer capítulo aborda la descripción de los problemas bajo estudio y establecerá los lineamientos a seguir para esta investigación. El capítulo 2, describe a detalle conceptos necesarios para comprender los problemas de secuenciación en máquinas paralelas no relacionadas y sus distintas variantes. También se presentan conceptos necesarios para entender los modelos de programación lineal entera mixta, y los algoritmos metaheurísticos propuestos. Por último, en este capítulo se

enumeran trabajos relacionados a los problemas en cuestión, con el propósito de dar contexto sobre el estado actual del arte respecto a esta disciplina.

Los capítulos 3 y 4 estarán dirigidos al problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza y al problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido, y minimización del makespan, respectivamente. Detallando en ambos, el problema específico a tratar, los modelos matemáticos y algoritmos metaheurísticos propuestos para resolver dichos problemas, la experimentación realizada y los resultados obtenidos.

Por último, en capítulo 5 se discuten las conclusiones obtenidas sobre la investigación presente, y se proponen nuevas líneas de investigación que puedan surgir a partir de lo presentado en esta tesis.

2. Marco teórico.

En este capítulo se presentarán los conceptos y fundamentos necesarios para comprender las metodologías de solución propuestas para los problemas bajo estudio. En primer lugar, se presentarán clasificaciones de los problemas de secuenciación según sus distintas características operativas con el propósito de brindar un mejor contexto a los problemas presentados. Posteriormente, se presentará una introducción a la programación matemática y se describirán algunos conceptos necesarios para facilitar el entendimiento de los modelos lineales propuestos. Después, se describirán características generales sobre metaheurísticas. Por último, se detallará información sobre trabajos en la literatura relacionados a los problemas de secuenciación que tengan características en común con los problemas presentados en esta tesis.

2.1. Clasificación de problemas de secuenciación.

Los problemas de secuenciación son un tipo de problema de optimización combinatoria en el que se tiene un conjunto de tareas que deben ser asignadas a una serie de procesadores (máquinas), de tal manera que se obtenga el mejor resultado posible para una función de valor (o función objetivo).

Graham, et al. (1979), presentan una metodología para clasificar problemas de secuenciación, describiendo la gran diversidad de variantes que puede tener el problema y codificándolos en un sistema simbólico compacto, en el cual contempla 3 principales características del sistema, que son: el tipo de entorno productivo (también conocido como entorno de máquinas), las características de operación (también conocidos como restricciones o características de las tareas), y el del objetivo del sistema (también conocido como función objetivo). Todas las

clasificaciones anteriormente mencionadas a su vez tienen sus propias subclasificaciones. Todas éstas serán detalladas a continuación.

2.1.1. Según el tipo de entorno productivo.

El entorno productivo describe cómo se relacionan las máquinas y las tareas dentro del sistema; describiendo si la tarea necesita ser procesada en una o múltiples etapas, y cuáles son las características de las máquinas que efectuarán dichas etapas. Para el caso de un sistema de una etapa, cada tarea debe ser procesada una sola vez. A continuación, se describen las configuraciones más comunes relacionadas a los problemas de secuenciación de etapa única:

- **Problema de máquina única:** considera el sistema más básico, donde una sola máquina procesa todas las tareas, y debido a que dicha máquina solo puede procesar una tarea a la vez, surge la necesidad de una secuencia; este problema tendrá tantas posibles soluciones como cantidad de permutaciones de dichas tareas (n!). Información adicional sobre este tipo de sistemas, se puede encontrar en el trabajo de *Gupta & Kyparisis (1987)*.

- **Problema de máquinas paralelas idénticas:** considera un sistema con múltiples máquinas, las cuáles son todas capaces de procesar cualquiera de las tareas con idéntico tiempo de procesamiento. En este sistema (y otros de etapa única con múltiples máquinas en paralelo), las tareas serán repartidas (asignadas) entre las distintas máquinas para ser procesadas, para cada máquina se formará una secuencia de procesamiento con sus respectivas tareas asignadas, y cada máquina ejecutará sus tareas de manera independiente, trabajando así todas al mismo tiempo (en paralelo). Información adicional sobre este tipo de sistemas se puede encontrar en el trabajo *de Cheng & Sin (1990).*

- **Problema de máquinas paralelas uniformes:** considera un sistema semejante al anterior, pero con la distinción de que las máquinas tienen distintos tiempos de procesamientos para resolver las tareas. Sin embargo, estos tiempos están relacionados entre sí por una constante. Se puede encontrar más información sobre este tipo de sistemas en el trabajo de *Blazewicz, Dror & Weglarz (1991).*

- **Problema de máquinas paralelas no relacionadas:** considera también un sistema con varias máquinas trabajando al mismo tiempo, y entre las cuáles se repartirán las tareas para ser procesadas y se determinará el orden en que cada máquina procesará sus correspondientes tareas. Sin embargo, en este sistema, cada máquina requiere un tiempo distinto en procesar cualquier tarea, y estos tiempos no pueden expresarse en términos de los tiempos de las otras máquinas. Se puede encontrar más información sobre este tipo de sistemas en el trabajo de *Li & Yang (2009).*

En los sistemas de múltiples etapas, cada trabajo necesita de la ejecución de múltiples procesos, es decir, debe ser atendida por varias máquinas: un ejemplo de esto podría ser un sistema de ensamblaje de vehículos, donde la creación de un vehículo específico sería considerada como una tarea, y las distintas piezas que se ensamblan se considerarían como las etapas. A continuación, se describen las configuraciones más comunes relacionadas a los problemas de secuenciación de etapas múltiples:

- **Problema de Open-shop**: en este problema, la ejecución de los múltiples procesos necesarios para completar un trabajo, se pueden ejecutar en cualquier orden, y todos los trabajos requieren pasar por la totalidad de etapas en el sistema. Más información sobre este tipo de sistemas se puede encontrar en el trabajo de *Anand & Panneerselvam (2015).*

- **Problema de *Flow-shop*:** este problema es similar al anterior, con la característica de que los múltiples procesos que se necesitan para ejecutar los trabajos tienen un orden establecido. Más información sobre este tipo de

sistemas se puede encontrar en el trabajo de *Komaki, Sheikh & Malakooti (2019).*

- **Problema de Job-shop:** en este problema se considera que no todos los trabajos necesitan ser procesados en todas las etapas disponibles en el sistema, sino en un subconjunto de éstas, único para cada trabajo. Más información sobre este tipo de sistemas se puede encontrar en el trabajo de *Chaudhry & Khan (2016).*

Esta última clase de problemas no serán estudiados a tanta profundidad en esta tesis. Sin embargo, se menciona debido a que configuraciones más complejas de estos sistemas pueden contener máquinas paralelas en sus múltiples etapas, y dichas etapas podrían estudiarse aisladamente como sistemas de máquinas paralelas. Ejemplos e información adicional sobre estos tipos de problemas puede encontrarse en el trabajo de *Linn & Zhang (1999)*, el de *Kyparisis & Koulamas (2006)* y el de *Ruiz & Vázquez-Rodríguez (2010).*

Adicionalmente, los sistemas podrían tener disponibilidad de máquinas, es decir, podría haber momentos del tiempo en que las máquinas no puedan trabajar, y dicho aspecto se deberá tomar en cuenta para la creación de la secuencia.

2.1.2. Según restricciones de operación.

Las restricciones de operación describen comportamientos especiales que pueden presentarse entre las tareas, en su interacción con las máquinas, e inclusive en la necesidad de recursos adicionales para su procesamiento.

Características especiales que se pueden presentar son las siguientes:

- **Permisividad de Interrupción:** el procesamiento de las tareas puede ser interrumpido y reanudado posteriormente. Se puede encontrar más información este tipo de sistemas en el trabajo *de Lawler & Labetoulle (1978).*

- **Tiempo de liberación:** Momento desde el cuál se puede iniciar a procesar una tarea. Se puede encontrar más información sobre este tipo de sistemas en el trabajo de *Centeno & Armacost (1997).*

- **Fechas de entrega:** define si las tareas del sistema tendrán asociado un momento del tiempo en el que deben ser terminadas. Más información sobre este tipo de sistemas se puede encontrar en el trabajo de *Gordon Proth & Chu (2002).*

- **Secuenciación *online*:** describe un sistema cuyas características de operación cambian en tiempo real, resultando en la necesidad de realizar ajustes a las secuencias de procesamiento durante la operación del sistema. El caso contrario sería un sistema de secuenciación *offline*, en el que las características del sistema son fijas durante toda la operación del sistema y se conocen *a* priori. Más información sobre este tipo de sistemas se puede encontrar en el trabajo de *Sgall (1998).*

- **Tiempos de preparación:** se presenta cuando las máquinas necesitan de un procedimiento especial para iniciar el procesamiento de una tarea. El tiempo requerido para éste podría ser fijo, depender de la máquina a la que la tarea fue asignada, o de quien fue la tarea anterior que fue ejecutada en la máquina (es decir, de la secuencia). Más información sobre este tipo de sistemas se puede encontrar en el trabajo de *Allahverdi, et al. (2008).*

2.1.3. Según el objetivo.

El objetivo del sistema describe, mediante métricas del desempeño, aquellas consecuencias resultantes de las decisiones tomadas que se desea mejorar. En los problemas de secuenciación de tareas, las métricas de desempeño típicamente suelen implicar el uso de tiempo, costos, o algún otro recurso del sistema. Estas métricas pueden tener muchas variantes, de las cuales las más comunes se describen a continuación:

- **Makespan:** representa tiempo en el que se termina de procesar la última tarea, representando así, el momento de tiempo en que el sistema completo queda libre para atender otro conjunto de tareas. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Reza Hejazi & Saghafian (2005).*

- **Tiempo de completación total:** representa la suma de los tiempos en el que salen las tareas del sistema, es una métrica relacionada con la eficiencia. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Sadfi, et al. (2005).*

- **Tiempo de completación con pesos (ponderaciones):** es una variante del problema de tiempo de completación total; en esta métrica los pesos pueden representar un costo asociado al tiempo en que las tareas permanecen en el sistema. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Schulz (1996).*

- **Tardanza total:** la suma de las tardanzas de todas las tareas, entendido esto como la cantidad de unidades de tiempo con la que se rebasó la fecha de entrega establecida, con respecto a la fecha en que realmente se entregó la tarea. Hace las veces de una métrica de servicio. Más información sobre

sistemas con este objetivo, puede encontrarse en el trabajo *de Potts & Van Wassenhove (1982).*

- **Tardanza máxima:** representa la tardanza de mayor valor de entre todas las tareas, esta métrica también está relacionada con medidas de servicio y minimizar su valor, resultará una atención más ecuánime entre las tareas o clientes. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Jackson (1955).*

- **Número de tareas tardías:** en esta métrica solo se contabilizan la cantidad de tareas tardías, sin importar que tan tardías estén. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Lee & Kim (2012).*

- **Tardanza con pesos:** representa una variante del problema de tardanza total. En esta métrica los pesos pueden representar alguna consecuencia asociada a la entrega tardía de una tarea, como lo puede ser un costo de penalización, el cuál podría ser distinta para cada tarea. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Abdul-Razaq, Potts & Van Wassenhove (1990).*

- **Tardanza más prontitud:** puede entenderse desde el concepto de justo a tiempo, en el que se pretende que toda tarea sea completada en el momento esperado (no antes y no después); esta métrica cuantifica la desviación absoluta entre la fecha de entrega establecida con respecto a la fecha de entrega real. Más información sobre sistemas con este objetivo, puede encontrarse en el trabajo de *Kedad-Sidhoum, Solis & Sourd (2008).*

En el trabajo de *Nagar, Haddock & Heragu (1995)* se muestra un estudio detallado sobre múltiples funciones objetivo y bi-objetivo. Más información sobre problemas de secuenciación, puede encontrarse en *Blazewicz, et al. (2019)*, *Leung (2004)* y *Kan (2012).*

2.1.4. Notación formal de los problemas.

La notación introducida por *Graham, et al. (1979)* cuenta con 3 parámetros principales, denotados de la forma $\alpha\ /\beta\ /\gamma$, donde α describe el entorno productivo, β las características de operación y γ describe el objetivo. Cada parámetro puede estar compuesto por varios elementos, los cuales a su vez pueden tomar distintos valores. Un resumen de las codificaciones más comunes se muestra en la Tabla 1, donde se muestra, para cada parámetro, que elemento(s) puede(n) conformarlo, y para cada elemento el vector de posibles valores que puede tomar, junto con una descripción en la que se muestra las descripciones correspondientes al vector de valores:

Tabla 1 : Descripción de la notación ***α / β / γ***

Parámetro	Elementos	Valores	Descripción
α	α1	$\{P, Q, R, F, J, O\}$	Este campo indica la configuración de las máquinas, y pueden ser **{paralelas idénticas, paralelas uniformes, paralelas no relacionadas, Flow Shop, Job Sop, Open shop}** respectivamente. Solo se omite para el caso de máquina única.
	α2	m	Indica el número de máquinas en el sistema (o número de etapas), en caso de omitirse se considera variable.
	α3	h_{il}	Describe los intervalos de disponibilidad de las máquinas, en caso de omitirse se considera disponibilidad total.
β	β1	$\{t - pmtn, pmtn\}$	Denotan interrupción del procesamiento de las tareas, siendo posible **{operación interrumpida, interrupción**

Parámetro	Elementos	Valores	Descripción
			arbitraria} respectivamente, en caso de omitirse se considera sin interrupción.
	β2	r_j	Denota el tiempo de liberación de las tareas, en caso de omitirse se consideran siempre disponibles.
	β3	d_j	Denota fechas de entrega para las tareas, en caso de omitirse se considerarán tareas sin fecha de entrega.
	β4	$online$	Denota un problema del tipo online, en caso de omitirse se considerará offline.
	β5	$\{s_{jk}, s_{ijk}\}$	Indica la necesidad de procesos de preparación, que pueden ser **{dependientes de la secuencia, dependiente de la máquina y secuencia}** respectivamente, en caso de omitirse se considera que no hay procesos de preparación.
γ	γ1	$\{C_{max}, \Sigma C_j, \Sigma w_j C_j, \Sigma T_j, T_{max}, \Sigma U_j, \Sigma w_j T_j, \Sigma(E_j + T_j)\}$	Indica el tipo de función objetivo **{makespan, tiempo de completación total, completación con pesos, tardanza total, tardanza máxima, numero de tareas tardías, tardanza con pesos, tardanza más prontitud}** respectivamente.

Para esta tesis, se estudiarán, en primer lugar, el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza total; el cual es un problema del tipo $R \mid d_j \mid \Sigma T_j$. En segundo lugar, se estudiará el problema de secuenciación de máquinas paralelas no relacionadas con tiempos de preparación dependientes de secuencia, recurso compartido, y minimización del makespan; el cual puede clasificarse como $R \mid s_{ijk} \mid C_{max}$ con recurso compartido evitando traslapes. Entre las varias técnicas para abordar dichos problemas, se encuentra la programación matemática, que será descrita a continuación.

2.2. Programación matemática.

De manera más coloquial, la programación matemática se puede entender como una herramienta para mejorar (optimizar) el resultado de un evento o sistema, que dependerá de las decisiones o acciones que se tomen en él y se hace de la siguiente manera: Primero, se establecen variables que representan las decisiones a tomar (conocidas como variables de decisión). Posteriormente, se crean funciones matemáticas que representarán el resultado a optimizar (conocida como función objetivo) y también las funciones que describirán los distintos comportamientos o reglas del sistema (conocidas como restricciones). Finalmente, mediante estas representaciones, se ejecutarán procedimientos que permitan determinar las decisiones adecuadas que optimicen la función objetivo deseada.

Un modelo lineal es un modelo de programación matemática en el que todas sus funciones están conformadas por operaciones derivables de orden 1, es decir, que pueden ser representadas mediante rectas en el plano cartesiano para problemas con solo 2 variables, planos en el espacio tridimensional para problemas con 3 variables, e hyper-planos en espacios multidimensionales para problemas de más variables.

A continuación, se mostrarán los conocimientos básicos para poder entender las propuestas de programación matemática realizadas en esta tesis, y se dará contexto de cómo esta herramienta ha sido utilizada en el ámbito de los problemas de secuenciación.

2.2.1. Introducción a la programación lineal.

De manera más formal podemos definir un modelo de programación matemática como un problema de la forma:

$$\max cx$$

$$Sujeto\ a: Ax \leq b$$

$$x \geq 0$$

Donde x es el vector columna de variables de decisión $x = (x_1..x_n)^T, x \in \mathbb{R}^n$, C es el vector fila de coeficientes de la función objetivo $c = (c_1..c_n)$, b el vector columna de lados derechos $b = (b_1..b_m)^T$, y la matriz de tamaño $m \times n$ de coeficientes de restricciones $A = (a_{ij}), i = 1..m, j = 1..n$, teniendo a n que representa la cantidad de decisiones a tomar, m representa la cantidad de restricciones. Este problema tendrá un conjunto de soluciones factibles $S \coloneqq \{x: Ax \leq b, x \geq 0\}$ dentro del cual existirá un vector x^* de tal manera que $cx^* \geq cx\ \forall\ x\ \in S$.

Lo que esta definición nos describe, expresado de manera más simple, es que, para un problema de maximización existe un conjunto S, también conocido como región factible, en la que están contenidas todas las soluciones x, las cuales deben de cumplir las distintas restricciones del problema definidas por los sistemas de ecuaciones $Ax \leq b\ y\ x \geq 0$, y de entre las cuales existe una solución x^*, la cual tiene un mayor valor en la función objetivo cx que cualquier otra solución x.

El conjunto de soluciones S puede presentar restricciones de integralidad para algunas, o todas las variables de decisión; lo que resultaría en un modelo lineal entero mixto (MILP) o un modelo lineal entero (ILP) respectivamente. Estos tipos de modelos presentan una dificultad especial al explorar su región factible, debido a una aparente "*discontinuidad"* que se presentan entre los distintos elementos de S. En la Figura 1 se ilustra para un modelo en $x \in \mathbb{R}^2$, con $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$ y $b = \begin{pmatrix} 5 \\ 4 \\ 3 \end{pmatrix}$, de manera gráfica, el conjunto S si ninguna variable de decisión se considera continua (Figura 1A), si solo una variable se considera continua (Figura 1B), y si todas las variables se consideran continuas Figura 1C); es evidente que para los últimos dos casos, se requiere de algoritmos con procesos enumerativos, como el algoritmo de ramificación y acotamiento, para resolver el modelo, el cual puede resultar en un incremento de la complejidad del modelo. Más información sobre programación entera en *Wolsey & Nemhauser(1999)* y *Conforti, Cornuéjols, & Zambelli (2014)*

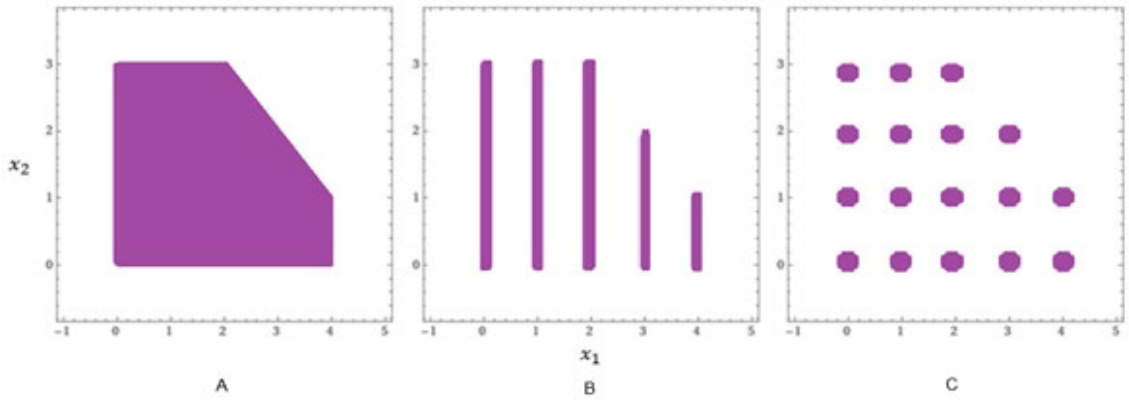


Figura 1: Región factible ilustrativa

Debido a la complejidad asociada a un modelo con variables enteras, es necesario implementar técnicas para expresar más adecuadamente un modelo. A continuación, se discuten algunas de éstas para expresar modelos de secuenciación:

2.2.2. Conceptos de modelación en problemas de secuenciación.

Al modelar problemas de secuenciación, existen varias maneras de expresar los distintos aspectos del sistema, desde las variables de decisión hasta las restricciones. Todas ellas pueden expresarse utilizando conceptos diferentes, los cuales pueden resultar más o menos eficientes según las características específicas del problema que se está enfrentando.

Los enfoques para presentar un modelo se presentan principalmente según como se concibe la secuencia; estos pueden ser enfoques de sucesión y precedencia, de secuencia directa, de posiciones, y del tiempo discreto. Dichos conceptos se detallan a continuación:

- En el enfoque de sucesión y precedencia, se establecen relaciones donde las tareas identifican a las demás, ya sea como sucesor o predecesor. Por ejemplo; la tarea A va antes de la B y la C, y la tarea B va antes de la C y después de la B, resultando en la secuencia A-B-C.

- En el enfoque de secuencias directas, se establecen las relaciones entre las tareas contiguas en la secuencia, es decir, que tarea va después de cual. Por ejemplo; la tarea B se ejecuta directamente después de la tarea A y la tarea C se ejecuta directamente después de la tarea B, resultando en la secuencia A-B-C.

- En el enfoque de tareas en posiciones, a cada tarea se le asigna un turno o posición en la secuencia. Por ejemplo: la tarea A es la primera en atenderse, la tarea B es la segunda en atenderse, y la tarea C es la tercera en atenderse.

- En el enfoque de tiempos discretos, se establece un intervalo de tiempo dentro del cual se establecerá la secuencia de tareas. Dicho intervalo es

dividido en unidades básicas discretas, y las tareas se controlan desde dichas unidades. Por ejemplo; la tarea A se iniciará en la unidad de tiempo 1.

Un trabajo más detallado sobre la aplicación de distintos enfoques de modelación a problemas de secuenciación en maquina única puede encontrarse en *Keha, Khowala & Fowler (2009)*. A modo de ilustración, se presentan los distintos enfoques aplicados al problema $1 \,//\, \Sigma w_j C_j$.

Considere n tareas a secuenciar que pertenecen a un conjunto de tareas $N = \{1,2,\ldots,n\}$, cada una contando con un tiempo de procesamiento p_j y con un peso o penalización w_j . Aplicando el enfoque de sucesión y precedencia, se definen las variables de decisión binarias X_{ij} que tomará el valor de 1 si la tarea i es antecesora a la tarea j, o 0 si la tarea i es sucesora de la tarea j, con $i,j \in N, i < j$. Adicionalmente se define la variable C_j, que representa el tiempo en que se completa la tarea j. El modelo entero mixto será:

$$\min \sum_{j \in N} w_j C_j \qquad \text{Eq.1}$$

$$C_i \leq C_j - p_j + \mu\left(1 - X_{ij}\right) \qquad \forall\, i,j \in N, i < j \qquad \text{Eq.2}$$

$$C_j \leq C_i - p_i + \mu\left(X_{ij}\right) \qquad \forall\, i,j \in N, i < j \qquad \text{Eq.3}$$

$$C_j \geq p_j \qquad \forall\, j \in N \qquad \text{Eq.4}$$

$$C_j \geq 0 \qquad \forall\, j \in N \qquad \text{Eq.5}$$

$$X_{ij} = \{1,0\} \qquad \forall\, i,j \in N, i < j \qquad \text{Eq.6}$$

Donde la Eq.1 calcula la función objetivo de manera típica, mientras que las Eq.2 y Eq.3 evitan el traslape entre tareas según las tareas sea sucesora o predecesora una de la otra. El grupo de ecuaciones Eq.4 evita que la primera tarea en la secuencia tenga tiempo de completación 0. Las Eq.5 y Eq.6 denotan la naturaleza de las variables y μ representa una constante grande. Esta concepción del problema cuenta con $(n^2 - n)/2$ variables binarias, n variables continuas y n^2 restricciones. Más información sobre este enfoque y el uso de sus variables puede encontrarse en el trabajo de *Queyranne & Wang (1991)* y *Queyranne & Schulz (1994).*

Para un modelo con un enfoque de secuencias directas se necesita de una tarea artificial 0 que servirá como inicio de la secuencia. Por lo que se considerará el conjunto $N_0 = \{N \cup 0\}$. Se considerará las variables de decisión X_{ij} que tomará el valor de 1 si la tarea i es antecesora directa de la tarea j, y el valor de 0 en caso contrario, con $i \in N_0, j \in N, i \neq j$. Las variables C_j tendrán el mismo concepto. La función objetivo y las naturalezas de las variables serán las mismas que las del modelo original, con lo que se conservarán las Eq.1 y Eq.5. Las restricciones por su parte serán distintas y se muestran a continuación:

$$C_i \leq C_j - p_j + \mu(1 - X_{ij}) \qquad \forall\, i, j \in N, i \neq j \qquad \text{Eq.7}$$

$$\sum_{j \in N} X_{0j} = 1 \qquad \text{Eq.8}$$

$$\sum_{i \in N_0, i \neq j} X_{ij} = 1 \qquad \forall j \in N \qquad \text{Eq.9}$$

$$\sum_{j \in N, i \neq j} X_{ij} \leq 1 \qquad \forall i \in N \qquad \text{Eq.10}$$

$$\sum_{\substack{h \in N_0 \\ i \neq h \neq j}} X_{hi} \geq X_{ij} \qquad \forall\, i, j \in N, i \neq j \qquad \text{Eq.11}$$

$$X_{ij} = \{1,0\} \qquad \forall\, i \in N_0, j \in N, i \neq j \qquad \text{Eq.12}$$

Donde, el conjunto de ecuaciones Eq.7 nuevamente tiene la función de evitar traslapes. La Eq.8 forzará la asignación de la tarea artificial como tarea inicial mientras que el grupo de ecuaciones Eq.9 asegura la asignación de todas las tareas, el conjunto de ecuaciones Eq.10 evita que una tarea tenga múltiples sucesores. El conjunto de ecuaciones Eq.11 es la restricción de conservación de continuidad en la secuencia, la Eq.12 es la naturaleza de la nueva variable. Esta concepción del problema cuenta con $n^2 - n$ variables binarias, n variables continuas y $2n^2 - n + 1$ restricciones. Más información sobre este enfoque y el uso de sus variables puede encontrarse en el trabajo de *Blazewicz, Dror & Weglarz (1991)* y *Dyer & Wolsey (1990).*

Para un modelo con un enfoque de tareas en posiciones, se considerará un conjunto de posiciones $L = \{1..n\}$, y se considerará las variables binarias X_{lj} que tomarán el valor de 1 si la tarea j se asigna a la posición l, 0 en caso contrario, con $l \in L, j \in N$. Dado la concepción de posiciones, los momentos en que se completen las actividades se conciben según la variable K_l, que representará el momento en que una posición complete el procesamiento de cualquiera que fuese su tarea asignada. El valor de dichas variables posteriormente será transferido a las variables C_j correspondientes. La Eq.1 se mantiene, al igual el conjunto de ecuaciones Eq.5. El resto de las restricciones necesarias son:

$$K_l \geq K_{l-1} + \sum_{j \in N} p_j X_{ij} \qquad \forall\, l \in L, l > 1 \qquad \text{Eq.13}$$

$$K_1 \geq \sum_{j \in N} p_j X_{ij} \qquad \text{Eq.14}$$

$$\mu\left(1 - X_{lj}\right) + C_j \geq K_l \qquad \forall\, l \in L, j \in \mathrm{N} \qquad \text{Eq.15}$$

$$\sum_{l \in L} X_{lj} = 1 \qquad \forall j \in N \qquad \text{Eq.16}$$

$$\sum_{j \in N} X_{lj} = 1 \qquad \forall l \in L \qquad \text{Eq.17}$$

$$K_l \geq 0 \qquad \forall\, l \in L \qquad \text{Eq.18}$$

$$X_{lj} = \{1,0\} \qquad \forall\, l \in L, j \in \mathrm{N} \qquad \text{Eq.19}$$

En este caso los conjuntos de ecuaciones Eq.13 y Eq.14 cumplen la función calcular los tiempos de completación de las posiciones, el conjunto de ecuaciones Eq.15 se encarga de transferir dichos valores a las variables de completación de las tareas, el conjunto de ecuaciones Eq.16 asegura que todas las tareas se ejecuten, mientras que el conjunto de ecuaciones Eq.17 asegura que en cada posición solo se asigne una tarea, y los conjuntos de ecuaciones Eq.18 y Eq.19 son las naturalezas de las nuevas variables. Este modelo tendría n^2 variables binarias, $2n$ variables continuas y $n^2 + 2n$ restricciones. Más información sobre este enfoque

y el uso de sus variable, puede encontrarse en el trabajo de *Lasserre & Queyranne (1992)* y *Queyranne & Schulz (1994).*

Por último, se considera el enfoque de tiempos discretos, para el cual se establece un rango de tiempo k para establecer la secuencia que, para el caso de máquina única, es conveniente establecer $k = \sum_{j \in N} p_j$. Se considerará el conjunto de periodos $K = \{1..k\}$ y se considerarán las variables binarias de decisión X_{qj}, que tomarán el valor de 1 si la tarea j es completada en el periodo q, o 0 en caso contrario, con $j \in N, q \in K, q \geq p_j$. El modelo entero mixto resultante para esta concepción del problema es el siguiente:

$$\min \sum_{j \in N} \sum_{q \in K} w_j q X_{qj} \qquad \text{Eq.20}$$

$$\sum_{q - p_j \leq h \leq q} X_{hj} \leq \mu(1 - X_{qj}) \qquad \forall\, j \in N, q \in K, q \geq p_j \qquad \text{Eq.21}$$

$$\sum_{q \in K, q \geq p_j} X_{qj} = 1 \qquad \forall j \in N \qquad \text{Eq.22}$$

$$X_{qj} = \{1,0\} \qquad \forall\, j \in N, q \in K, q \geq p_j \qquad \text{Eq.23}$$

En este caso la Eq.20 calcula la función objetivo en términos de las variables de tiempo discreto, las Eq.21 evitan traslape entre el procesamiento de diferentes tareas, las Eq.22 aseguran la ejecución de las tareas y el conjunto de ecuaciones Eq.23 denota la naturaleza de las variables. Este modelo tendría aproximadamente de nk variables binarias, y menos de $nk + n$ restricciones. Más sobre este enfoque

y el uso de sus variables puede encontrarse en el trabajo de *Abdul-Razaq, Potts & Van Wassenhove (1990)* y *Sousa & Wolsey (1992).*

Es notable cómo los distintos enfoques del problema resultan en diferentes modelos con sus diferentes características, siendo algunos más aptos que otros para lidiar con características específicas de los problemas de secuenciación. En este caso, se puede observar cómo las ecuaciones Eq.2, Eq.3, Eq.7, y Eq.21 son todas necesarias para evitar el traslape de tareas en sus enfoques correspondientes, mientras el enfoque de posiciones no necesita evitar esto de manera directa. En cambio, evita el uso inadecuado de las posiciones en la Eq.17. Se pueden mencionar otros ejemplos de estas distinciones como en el caso del enfoque de variables de secuencia directa, que suele ser el más apto para tratar con problemas que tengan tiempos de preparación dependientes de la secuencia, El enfoque de tiempos discretos puede facilitar el tratar con recursos adicionales cuya disponibilidad varia a través del tiempo, y el enfoque de sucesión y precedencia puede ser útil en problemas donde se tengan restricciones de secuencia, es decir, no se permite procesar una tarea sin antes haber completado otra.

A pesar de existir distintos enfoques que puedan facilitar el modelar ciertos entornos productivos, no todos son susceptibles a ser descritos mediante modelos lineales, como puede ser el caso de los problemas online, como se puede ver los trabajos de *Wenzelburger & Allgöwer (2019)* y de *Cho & Easwaran (2020).* Adicionalmente, la resolución de los modelos puede ser computacionalmente demandante. En el trabajo de *Balas (1985)* se realiza un estudio sobre las caras de las envolturas convexas asociadas a los problemas de secuenciación, en el trabajo de *Lenstra, Kan & Brucker (1977)* se realiza un estudio detallado sobre la complejidad de distintos problemas de secuenciación, mientras que el trabajo de *Mokotoff (2001)* se discute para variantes específicas del problema de secuenciación, que dichos problemas son de complejidad no polinomial dura. Debido a estas razones, es necesario la implementación de otras metodologías que puedan brindar buenas soluciones en tiempos computacionales razonables.

Para esta tesis, el problema $R\ /\ d_j\ /\ \Sigma T_j$ será abordado con un modelo lineal entero mixto basado en el enfoque de posiciones, mientras que el problema $R\ /\ s_{ijk}\ /\ C_{max}$ con recurso compartido sin traslape será abordado con un modelo lineal basado en el enfoque de secuencia directa.

2.3. Metaheurísticos.

En el ámbito de la optimización, la solución óptima no siempre puede ser encontrada en un tiempo razonable. Esto debido a que las herramientas que aseguran su obtención no siempre pueden ser aplicadas a todos los problemas que se presentan en la práctica, como se ha discutido en trabajos desde *Graves (1981)* hasta *Parente, et al. (2020).* Técnicas como programación dinámica, optimización por gradiente, o algoritmos de ramificación y poda tienen criterios estrictos sobre las características de los problemas que pueden abordar, o en su defecto, crecen exponencialmente en complejidad resultando en tiempos de resolución inviables.

Las técnicas metaheurísticas son de aplicación general, es decir, no específicas para ningún problema, y se presentan más bien como una serie de conceptos o premisas cuya lógica pretende guiar la exploración del espacio de soluciones. Las metaheurísticas pueden apoyarse de métodos heurísticos y otros algoritmos específicos al problema en cuestión, como los mostrados en *Panwalkar & Iskander (1977),* para la creación de soluciones iniciales o para efectuar la exploración del espacio de soluciones, imponiendo sobre estas metodologías reglas o procedimientos adicionales que hacen más eficiente la búsqueda y facilitan la obtención de buenas soluciones. Cabe resaltar que dichos mecanismos podrían implicar procedimientos aleatorios e iteraciones constantes.

En los procesos metaheurísticos encontramos el concepto de vecindario, el cual, es una región específica del espacio de soluciones; es decir, un conjunto de

soluciones que se pueden considerar cercanas entre sí al poseer todas, alguna o varias características en común. En los procedimientos que puede implementar un algoritmo metaheurístico se pueden identificar enfoques relacionados a la exploración del espacio de soluciones y que están ligados al concepto del vecindario. En particular se definen dos enfoques principales para realizar dicha exploración, y estos son:

- **Intensificación:** en este enfoque se explora de manera detallada, e inclusive exhaustiva, un vecindario de soluciones en específico que se presume podría contener una buena solución. Esto tiene el propósito de alcanzar soluciones de mejor valor.

- **Diversificación:** en este enfoque se realizan procedimientos que amplían el espacio de búsqueda de soluciones a nuevas regiones del espacio de búsqueda. Esto se hace con el propósito de evitar estancarse en una región cuyas soluciones óptimas locales no sean de buena calidad.

Los algoritmos metaheurísticos tienen 2 clasificaciones principales: métodos de solución única y métodos basados en poblaciones, los cuales se describirán más a detalle a continuación. Una descripción aún más detallada se puede encontrar en el trabajo de *Boussaïd, Lepagnot, & Siarry (2013).*

2.3.1. Metaheurísticos de solución única.

También conocidos como métodos de trayectoria, estos métodos trabajan con una única solución del problema, la cual se va modificando y mejorando durante sus iteraciones, estableciendo así, una trayectoria en la cual se va desplazando la solución, y la cual, idealmente, apunta a mejores soluciones o a vecindarios prometedores. Algunos de los métodos más comunes en esta categoría son:

- **Recocido simulado:** inspirado en procesos metalúrgicos, en los cuales un metal caliente es enfriado de manera controlada para así obtener un material estable (estado de mínima energía). Esta metodología establece de manera análoga, en un problema de optimización, mecanismos de aceptación o rechazo de nuevas soluciones, basados en umbrales probabilísticos, según ciertos parámetros que hacen las veces de la “temperatura”; estos procesos de aceptación y rechazo resultan tanto en mecanismos de diversificación e intensificación, puesto que el algoritmo acepta con más facilidad nuevas soluciones durante las iteraciones iniciales del algoritmo, sin importar su calidad, así diversificando, y se vuelve más estricto a medida que el algoritmo avanza, así intensificando. Una descripción aún más detallada se puede encontrar en el trabajo de *Suman & Kumar (2006).*

- **Búsqueda tabú:** este algoritmo utiliza un proceso de aprendizaje formando estructuras de memoria, es decir, listas que registran movimientos de búsqueda realizados recientemente. Esto con el propósito de evitar dichos movimientos, pudiendo así escapar de óptimos locales y guiando la búsqueda hacia nuevos vecindarios. La temporalidad de la memoria controla la diversidad e intensificación de las soluciones, donde una lista que almacena y prohíbe ciertos movimientos durante muchas iteraciones, resultará en una búsqueda diversa, mientras que una lista con memoria de un plazo muy corto explorará a profundidad una región pequeña, resultado así en intensificación. Una descripción aún más detallada se puede encontrar en el trabajo de *Glover (1997).*

- **Procedimiento de búsqueda adaptativa aleatorio codicioso:** también conocido como GRASP por sus siglas en inglés, es un procedimiento multi-arranque, es decir, en cada iteración se ejecuta el mecanismo con el que se crea la solución inicial, lo cual brinda diversidad a la búsqueda, y sobre dichas soluciones se ejecutan mecanismos de exploración del vecindario de dicha solución, dando así intensificación a la búsqueda. Este algoritmo almacenará y presentará la mejor solución encontrada durante cualquiera de sus

iteraciones. Más sobre esta técnica, puede encontrarse en el trabajo de *Resende & Ribeiro (2010)*

- **Búsqueda por entornos variables:** en esta metodología se utilizan mecanismos para establecer múltiples vecindarios, posteriormente se seleccionan de manera aleatoria para ser explorados. Una vez explorado un vecindario se procede a explorar otro de los establecidos, sustituyendo la mejor respuesta encontrada hasta el momento en caso de encontrar una mejora. Este procedimiento brinda intensificación al explorar a profundidad cada vecindario y diversidad al establecer los múltiples de estos. Mayor información sobre esta técnica puede encontrarse en el trabajo de *Hansen & Mladenović (2001).*

- **Búsqueda local Iterada:** También conocido como ILS por sus siglas en inglés, genera una nueva solución para establecer un nuevo vecindario de búsqueda, perturbando la última solución encontrada; es decir, alterando aleatoriamente dicha solución. Este procedimiento pretende dar diversidad o intensidad a la búsqueda, según los criterios para aceptar la solución perturbada; donde criterios estrictos que exijan mejoras en la solución, resultarán en intensificación, mientras que criterios laxos que permitan soluciones peores, resultarán en diversificación. Más información sobre esta técnica puede encontrarse en el trabajo de *Lourenço, Martin & Stützle (2019).*

2.3.2. Metaheurísticos poblacionales.

Estos métodos trabajan con una multitud de soluciones, a los cuales se le suele llamar población; dichas poblaciones suelen ser manipuladas para generar nuevas soluciones, las cuales pueden incorporarse a la población original o sustituir miembros de ella. Los mecanismos de manipulación de las soluciones y generación de nuevas suelen tener como propósito la intensificación en la exploración de

soluciones, mientras que los mecanismos para seleccionar y mantener las poblaciones suelen procurar la diversificación de soluciones. Estas metodologías suelen tener inspiración en procesos naturales. Estos métodos pueden clasificarse en 2 principales categorías:

- **Computación evolutiva:** son algoritmos inspirados por los principios de la evolución, en la que los miembros más aptos de una población heredan sus características a la siguiente, quienes a su vez transforman dichos genes, posiblemente resultando en miembros más aptos, y con los cuales se realiza un proceso de selección natural para generar una nueva población y repetir el ciclo. Estos conceptos se aplican análogamente en el ámbito de la optimización, donde la aptitud de los miembros de la población de soluciones se mide mediante la función objetivo, se identifican las características de las mejores soluciones (a las cuales se les denomina padres), y éstas se transfieren a un conjunto de nuevas soluciones candidatas (a las que se les denomina hijos), a las cuales se les establecen cambios adicionales en sus características (denominado mutación), para posteriormente realizar el proceso de selección (nueva generación) e iterar nuevamente. Entre estos algoritmos podemos encontrar ejemplos como lo son los algoritmos genéticos, los algoritmos evolutivos, la búsqueda dispersa, re-encadenamiento de trayectorias, entre otros Una descripción aún más detallada se puede encontrar en el trabajo de *Lambora, Gupta & Chopra (2019)*

- **Algoritmos de inteligencia colectiva o de enjambre:** son algoritmos inspirados en mecanismos de interacción entre agentes de grupos sociales encontrados en la naturaleza, como parvadas de aves o colonias de insectos. En estos algoritmos se establecen analogías entre los comportamientos de los individuos del grupo y procesos de exploración de la región factible, teniendo así agentes (procesos de creación y modificación de soluciones), los cuales comunican entre sí características y parámetros sobre las soluciones encontradas en su exploración y modificando su conducta en

base a la información y retroalimentación del grupo. Entre estos algoritmos podemos encontrar ejemplos como lo son los algoritmos de optimización de colonia de hormigas, algoritmo de colonia de abejas, y optimización de enjambre de partículas. Más información sobre esta técnica, puede encontrarse en el trabajo de *Rajakumar, Dhavachelvan & Vengattaraman (2016).*

Para esta tesis, el problema $R \mid d_j \mid \Sigma T_j$ será abordado con un algoritmo metaheurístico de búsqueda local iterada, mientras que el problema $R \mid s_{ijk} \mid C_{max}$ con recurso compartido sin traslape, será abordado con un algoritmo metaheurístico de búsqueda por entornos variables. Una descripción aún más detallada sobre metaheurísticas se puede encontrar en el trabajo de *Potvin & Gendreau (2018)*, *Rafael, Pardalos & Resende (2018)* y *Malik, et al. (2021)*

2.4. Literatura relacionada.

En esta sección se discuten trabajos sobre el problema de secuenciación, haciendo énfasis en aquellos que tengan características en común con los problemas que se abordan en este estudio. Si algún trabajo contiene más de una característica que sea de interés, se mencionará solamente en una sección, aclarando cuales son las características adicionales que podrían hacerlo meritorio de mencionar en otras secciones.

2.4.1. Secuenciación con minimización de la tardanza y sus variantes.

En el trabajo de *Emmons (1969)* se estudia el problema de tipo $1 \mid d_j \mid \Sigma w_j T_j$, donde los pesos están dados por una función no lineal. En este trabajo se

establecen propiedades que debe de cumplir una solución óptima para el problema en cuestión, se analizan los efectos del intercambio de tareas en cuanto a la exploración del espacio de soluciones, y se explora el uso de reglas de secuenciación como SPT (shortest procesing time) y EDD (Earliest due date) para establecer vecindarios de búsqueda adecuados. Adicionalmente se propone un algoritmo heurístico que contempla las ideas anteriormente mencionadas y una exploración ramificada del espacio de soluciones. En este trabajo no se hace una experimentación extensiva, pero, se muestran soluciones para problemas de hasta 10 tareas.

En el trabajo de *Ho & Chang (1991)* se aborda inicialmente el problema de tipo $1\ /d_j/\ \Sigma T_j$, describiendo heurísticas comunes como lo son EDD, SPT; y proponiendo una nueva heurística que combina conceptos basados en las anteriormente mencionadas junto con búsquedas locales basadas en intercambios de tareas entre posiciones vecinas inspirada en el algoritmo de ordenamiento de burbujas. Adicionalmente, extiende los conceptos del algoritmo propuesto para ser aplicables al problema $P\ /d_j/\ \Sigma T_j$, añadiendo a los procedimientos una regla de balanceo de carga para la asignación de las tareas a las máquinas. Finalmente se muestra experimentación en la que se abordan instancias de hasta 15 tareas y 2 máquinas.

En el trabajo de *Azizoglu & Kirca (1998)* se estudia el sistema $P\ /d_j/\ \Sigma T_j$, en el cual se hace una caracterización detallada de las propiedades asociadas a una secuencia óptima, y basado en esto se proponen un algoritmo con solución inicial basadas en reglas de secuenciación (SPT y EDD) y que efectúa la exploración de la región factible mediante el método de ramificación y acotamiento, el cual limita la exploración de ramas basado en las propiedades que debería tener la solución óptima. Finalmente se muestra experimentación en la que se abordan instancias de hasta 15 tareas y 3 máquinas.

En el trabajo de *Liaw, et al. (2003)* se estudia el sistema $R\ /d_j/\ \Sigma w_j T_j$, donde se caracterizan las soluciones óptimas, se establecen reglas de dominancia, se

establecen heurísticas para la asignación de tareas a máquinas y se establecen cotas inferiores al problema; esto con el propósito de ser aplicado al proceso de exploración de soluciones del algoritmo de ramificación y acotamiento. Finalmente, se muestra experimentación en la que se abordan instancias de hasta 18 tareas y 4 máquinas.

En el trabajo de *Tanaka & Araki (2008)* se estudia el sistema $P\ /d_j/\ \Sigma T_j$, donde se presenta un modelo de programación lineal al cual se le aplican relajaciones lagrangianas, estrategias de dominación a respuesta parciales del problema, se establecen cotas superiores e inferiores a las soluciones y reglas para la ejecución del algoritmo de ramificación y acotamiento, con el propósito de reducir las ramificaciones efectuadas. Finalmente se muestra experimentación en la que se abordan instancias de hasta 25 tareas y 10 máquinas.

En el trabajo realizado por *De CM Nogueira, et al. (2014*) se estudia el sistema $R\ /d_j, \mathrm{s_{ijk}}/\ \Sigma(E_j + T_j)$, para el cual se establece un modelo de programación lineal entero mixto basado en variables de secuencia y también varios algoritmos metaheurísticos inspirados en GRASP, en los cuales se crean soluciones iniciales de manera codiciosa, al agregar elementos a una solución parcial hasta formar una solución factible, la cual se usa como punto de partida para explorar vecindarios locales; las distintas variantes del algoritmo incorporan conceptos de re encadenamiento de trayectorias y de ILS. Finalmente se muestra experimentación en la que se abordan instancias de hasta 30 tareas y 5 máquinas para el modelo lineal, y de hasta 100 tareas y 30 máquinas para los algoritmos propuestos.

En el trabajo de *Cheng & Huang (2017)* estudian el sistema $R\ /d_j/\ \Sigma(E_j + T_j)$ con máquinas dedicadas, en el cual, no todas las máquinas son capaces de ejecutar todas las tareas. Propone un modelo entero mixto basado en variables de secuencia y un algoritmo genético en el cual establece secuencias candidatas para crear la población de soluciones necesaria. Finalmente se muestra experimentación en la que se abordan instancias de hasta 15 tareas y 5 máquinas.

2.4.2. Secuenciación con recursos adicionales.

En el trabajo de *Ventura & Kim (2000)* se estudia sistema $P\ //\ \Sigma(E_j + T_j)$, en el que se le asocia ambas desviaciones con un costo y se requiere del uso de recursos adicionales para la ejecución de las tareas, los cuales son visualizados como elementos disponibles en cantidades conocidas a través del tiempo. En este trabajo se propone un modelo binario basado en variables de tiempos discretos, que posteriormente es reformulado como un modelo de asignación generalizada. En este trabajo no se muestra experimentación extensiva, pero en cambio se muestra la aplicación del algoritmo a manera de demostración para un problema de 5 tareas y 2 máquinas.

En el trabajo de *Ruiz & Andrés-Romano (2011)* se estudia el sistema $\mathrm{R}\,/\mathrm{s_{jk}}/\,\Sigma C_j$ en el que se consideran recursos compartidos los cuales pueden disminuir los tiempos de preparación si se invierte una cantidad adicional de estos; debido a esto, en la función objetivo también se incorpora la cantidad de recursos utilizados, resultando en un modelo bi-objetivo. Para este problema, se propone un modelo de programación entera mixta y varias heurísticas basadas en reglas de secuenciación las cuales compara entre sí. En este trabajo no se muestra experimentación extensiva, pero en cambio, se muestra la aplicación del algoritmo a manera de demostración para un problema de 10 tareas y 5 máquinas.

En el trabajo de *Fanjul-Peyro, Perea & Ruiz (2017)* se estudia el sistema $R\ //\ C_{max}$ con recursos adicionales, para el cual, muestran 2 modelos matemáticos; un modelo de programación lineal entera mixta basado en metodologías clásicas de tiempos discretos, y un modelo inspirado en el problema de empaquetamiento de tiras, en el cual se visualiza al límite de recursos en el tiempo como el ancho de la tira y el largo de la tira representará el makespan. Adicionalmente se muestran varias metaheurísticas para la asignación y fijación de tareas. En este trabajo no se

muestra experimentación extensiva, pero en cambio se muestra la aplicación del algoritmo a manera de demostración para un problema de 30 tareas y 6 máquinas.

En el trabajo de *Villa, Vallada & Fanjul-Peyro (2018)* se estudia el sistema $R\ //\ C_{max}$ con recursos adicionales escasos. Abordan dicho problema mediante 2 heurísticas distintas. En la primera proponen una metodología basada en construcción de soluciones basada en los recursos, en la cual se establecen búsquedas locales que siempre mantienen una solución factible. En la segunda muestran una heurística que da prioridad a la asignación codiciosa de tareas durante sus procesos de exploración de soluciones, sin importar si la solución resultante es o no factible en términos de los recursos adicionales disponibles, para después establecer mecanismos que reparan la solución, haciéndola factible. En este trabajo se muestra una experimentación extensiva, en la que se realiza comparativas con varias otras metodologías existentes en la literatura, abordando instancias de hasta 350 tareas y 30 máquinas.

En los trabajos de *Vlk, Novak & Hanzalek (2019)* y *Vlk, et al. (2019)* se estudia el sistema $P\ /s_{ijk}/\ C_{max}$ con un recurso compartido y procesos de preparación para los cuales no puede haber traslapes; adicionalmente se considera una asignación previa de las tareas a las máquinas, quedando solo por decidirse la secuencia de cada máquina. En ambos trabajos se muestra un modelo de programación lineal entera mixta basado en variables de posición y una aplicación basada en programación por restricciones, teniendo 3 variantes en el primer trabajo, y presentando 2 variantes adicionales en el segundo, así como un algoritmo heurístico basado en procesos de descomposición. En ambos trabajos se muestra experimentación para instancias de hasta 50 tareas por máquina y 50 máquinas.

En el trabajo de *Fanjul-Peyro (2020)* se estudia el sistema $R\ /s_{ijk}/\ C_{max}$ con recursos adicionales limitados, los cuales pueden estar asociados a la ejecución de la preparación, del procesamiento, o de ambos. En este trabajo se presenta un modelo de programación lineal entera mixta basada en variables de secuencia directa, que incorpora conceptos e ideas relacionadas al problema de

empaquetamiento de tiras y, un algoritmo de 3 fases, en el que primero se asignan tareas a máquinas, después se establecen las secuencias de cada máquina, y finalmente se verifica el uso de recursos. Finalmente se muestra experimentación en la que se abordan instancias de hasta 50 tareas y 8 máquinas.

2.4.3. Secuenciación con tiempos de preparación.

En el trabajo de *Lee & Pinedo (1997)* se estudia el sistema $P\ /d_j\ , s_{ijk}/\ \Sigma w_j T_j$ para el cual se propone un algoritmo metaheurístico de 3 fases; en la primera fase se analiza y caracteriza la instancia a abordar, en la segunda fase se crea una solución inicial mediante reglas de secuenciación, en las cuales, se incorporan reglas y metodologías determinadas por la caracterización de la instancia y en la tercera fase se aplica la metaheurística de recocido simulado. En este trabajo no se muestra experimentación directa sobre la metodología completa; en cambio se muestra experimentación realizada para la afinación de parámetros.

En el trabajo de *Radhakrishnan & Ventura (2000)* se estudia el sistema $P\ /d_j\ , s_{ijk}/\ \Sigma\left(E_j + T_j\right)$, para el cual se presenta un modelo de programación lineal entera mixta basado en variables de secuencia directa, y la aplicación de un algoritmo de recocido simulado, el cual toma la solución inicial de una heurística codiciosa que crea listas de candidatos para la asignación a una solución parcial basada en la regla de secuenciación y tiempos de completación más próximos; adicionalmente implementa mecanismos de generación de vecindarios de búsqueda basados en intercambios de tareas según métricas de aptitud para identificar y seleccionar los mejores candidatos. Finalmente se muestra experimentación en la que se abordan instancias de hasta 10 tareas y 3 máquinas.

En el trabajo de *Logendran, McDonell & Smucker (2007)* se estudia el sistema $R, h_{il}\ /r_j\ , d_j\ , s_{ijk}/\ \Sigma w_j T_j$, para el cual se proponen 6 diferentes algoritmos

de búsqueda tabú resultantes de establecer variaciones en los distintos parámetros del algoritmo como lo son el tamaño de las listas, el tiempo que los elementos permanecen en la memoria, y el manejo de la memoria de corto y largo plazo. Las búsquedas locales se realizan mediante movimientos de intercambio y reinserción de tareas. Para las distintas variantes del algoritmo de búsqueda tabú se crean soluciones iniciales basadas en distintas reglas de secuenciación. En este trabajo se presenta experimentación extensiva para instancias de hasta 7 tareas y 4 máquinas.

En el trabajo de *Vallada & Ruiz (2011)* se estudia el sistema $R\ /s_{ijk}/\ C_{max}$ en donde se presenta un modelo de programación lineal entera mixta, basada en variables de secuencia directa junto con 4 variantes de un algoritmo genético, en el que incorpora conceptos de otras heurísticas como la inserción múltiple para crear la población inicial, así como búsquedas locales basadas en intercambio e inserción exhaustiva de tareas entre máquinas. Finalmente se muestra experimentación en la que se abordan instancias de hasta 250 tareas y 30 máquinas.

En el trabajo de *Vallada & Ruiz (2012)* se estudia el sistema $R\ /s_{ijk}/\ \Sigma(E_j + T_j)$ para el cual se presentan dos modelos de programación lineal entera mixta, basados en variables de secuencia directa, pero siendo el segundo una reformulación que incluye variables auxiliares de asignación. Adicionalmente se propone un algoritmo genético que incluye procedimientos de inserción de tiempos ociosos en el sistema como mecanismo para reducir los tiempos de prontitud. Finalmente se muestra experimentación en la que se abordan instancias de hasta 250 tareas y 30 máquinas.

En el trabajo de *Tran, Araujo & Beck (2016)* se estudia el sistema $R\ /s_{ijk}/\ C_{max};$ en este trabajo se muestra un modelo de programación lineal entera mixta, el cual, es incorporado a un método exacto de descomposición en el que primero se asigna tareas a máquinas y posteriormente se crea la secuencia para cada máquina, adicionado a esto, aplica un procedimiento de ramificación y verificación para seleccionar y resolver los sub-problemas generados; para este último,

introduce conceptos relacionados al problema del agente viajero para hacer una representación de los sub-problemas emergentes. Finalmente se muestra experimentación en la que se abordan instancias de hasta 60 tareas y 5 máquinas.

En el trabajo de *Fanjul-Peyro, Ruiz & Perea (2019)* se estudia el sistema $R\ /s_{ijk}/\ C_{max}$ para el cual se propone un modelo de programación lineal entera mixta basada en variables de sucesión directa, el cual, también es aplicado en un algoritmo de descomposición y ramificaciones que mejora su desempeño mediante la inclusión de desigualdades válidas al modelo. Finalmente se muestra experimentación en la que se abordan instancias de hasta 1000 tareas y 8 máquinas.

2.4.4. Secuenciación con minimización del makespan.

En el trabajo de *Hariri & Potts (1991)* se estudia el problema $R\ //\ C_{max}$. En este trabajo se proponen varios algoritmos heurísticos de 2 fases, los cuales comparten el siguiente comportamiento; en la primera fase, se crea una asignación parcial de tareas a máquinas, realizada por un modelo de programación lineal que considera una versión relajada del problema, quedando para la segunda etapa, la asignación de las tareas faltantes, las cuales son integradas a la respuesta parcial mediante métodos heurísticos existentes en la literatura, mismos que toman en consideración información relevante obtenida del modelo lineal. Resuelve instancias de hasta 100 tareas con 50 máquinas.

En el trabajo de *Min & Cheng (1999)* se estudia el problema $P\ //\ C_{max}$ el cual es abordado mediante 3 metodologías; la primera es una heurística basada en la regla de secuenciación “tiempos de procesamientos más largos”, la segunda es un algoritmo genético de codificación binaria para los genes de sus individuos, y la tercera un algoritmo de recocido simulado con procesos de intercambio de tareas e inversión de secuencias. En este trabajo se realiza experimentación con el principal

propósito de comparar estas 3 metodologías, abordando instancias de hasta 30 tareas y 10 máquinas.

En el trabajo de *Yang & Yang (2010)* se estudia el problema $1 // C_{max}$ con actividades de mantenimiento variables y efecto de envejecimiento, el cual resulta en un incremento en los tiempos de procesamiento para aquellas tareas que estén al final de la secuencia, justificado por el desgaste de los equipos que realizan dicho procesamiento, y el cual puede ser prevenido mediante las actividades de mantenimiento. Para este sistema se presenta un modelo de programación lineal basado en variables de posición. En este trabajo no se muestra una experimentación extensiva, pero en cambio se analizan varias variantes del problema según distintos valores asociados a los parámetros de la instancia.

En el trabajo de *Avalos-Rosales, Alvarez & Angel-Bello (2013)* se estudia el sistema $R\ /s_{ijk}/\ C_{max}$ para el cual se muestran varios modelos de programación lineal entera mixta en los cuales se establecen distintas formas de calcular el makespan y se incluyen variables de asignación auxiliares continuas, permitiendo que los modelos sean resueltos más eficientemente por el algoritmo de ramificación y acotamiento. Adicionalmente proponen un algoritmo multi-arranque cuyos procesos de mejora integran conceptos de la metaheurística de búsqueda de vecindario variable. Finalmente se muestra experimentación en la que se abordan instancias de hasta 250 tareas y 30 máquinas.

En el trabajo de *Gedik, et al. (2018)* se estudia el sistema $\mathrm{R}\ /\mathrm{s_{ijk}}/\ \mathrm{C_{max}}$ para el cual se muestran un modelo de programación lineal entera mixta el cual es resuelto mediante metodologías de programación por restricciones y técnicas de ramificación en las que se prioriza la asignación de tareas a máquinas. Finalmente se muestra experimentación en la que se abordan instancias de hasta 11 tareas y 6 máquinas.

2.5. Aportaciones a la literatura.

Las contribuciones de esta tesis a la literatura son las siguientes:

i) Se propondrá, para el problema de secuenciación en máquinas paralelas no relacionadas con minimización de la tardanza total, un modelo de programación lineal entera mixta basado en variables de posición que puedan resolver instancias de mayor tamaño que los modelos actuales publicados en la literatura. Adicionalmente, se propondrá un algoritmo metaheurístico que permita obtener soluciones de alta calidad en tiempos computacionales competitivos.

ii) Se trabajará un problema que no ha sido estudiado en la literatura; el problema de máquinas paralelas no relacionadas con minimización de makespan, tiempos de preparación dependientes de la secuencia y recurso compartido. Para dicho problema se propondrán y comparan diferentes formulaciones matemáticas, así como se desarrollará un algoritmo metaheurístico que permita obtener soluciones de alta calidad en tiempos computacionales competitivos.

3. Problema de secuenciación de máquinas paralelas no relacionada con minimización de tardanza total.

La motivación de abordar el problema de máquinas paralelas surge de situaciones particulares que se presentan en la industria donde la adquisición de equipos sucede de manera discontinua como una respuesta al crecimiento de la industria o al aumento de la demanda. Esto puede reflejarse en la obtención de equipos con características diferentes, resultando en capacidades productivas inconsistentes; como se mencionó en anteriores capítulos, un ejemplo de esto se observa en las industrias de prototipado mediante impresión 3D, que, al ser una tecnología relativamente nueva, constantemente se presentan en el mercado nuevos equipos de impresión con características nuevas. Sin embargo, los equipos anteriores siguen siendo totalmente funcionales, resultando en una capacidad de producción instalada con características mixtas en los distintos equipos. Estos sistemas presentan dificultades adicionales en cuanto a la planeación de la producción, la cual puede ser más compleja para ciertos objetivos. Aunque existen varias métricas de interés para dichos problemas de secuenciación, la tardanza se vuelve crítica, ya que está directamente relacionada con la satisfacción del cliente y se ve impactada en los sistemas de máquinas paralelas no relacionadas debido a su inconsistencia en las capacidades productivas.

En capítulos anteriores se describió de manera más general el problema de secuenciación en máquinas paralelas no relacionadas con minimización de tardanza total, y se mencionó brevemente las características concernientes de este problema. En este capítulo se describe a detalle y de manera más formal el problema $R\ /d_j/\ \Sigma T_j$, el cual presenta una complejidad distinta a otras variantes del problema de secuenciación *(Sen, Sulek & Dileepan 2003)*, así como el modelo de programación lineal entera mixta y el algoritmo metaheurístico propuestos para abordar dicho problema.

3.1. Definición formal del problema.

La siguiente notación será utilizada para describir el problema $R\ /d_j/\ \Sigma T_j$:

- Existen n tareas a resolver, las cuales pertenecen al conjunto N; dichas tareas serán trabajadas por cualquiera de las m máquinas paralelas disponibles, las cuales pertenecen al conjunto M.
- Todas las máquinas están siempre disponibles, siendo cada una capaz de atender una sola tarea a la vez. Una vez que una máquina inicia el procesamiento de una tarea, no puede ser interrumpida.
- Todas las tareas están disponibles en todo momento para ser procesadas y no existen restricciones de precedencia entre tareas.
- Cada tarea $j \in N$ tiene un tiempo de procesamiento p_{ij} asociado a la máquina $i \in M$.
- Cada tarea j tiene una fecha de entrega d_j
- En caso de que el tiempo C_j en que se completa la tarea j sea posterior a su fecha de entrega correspondiente d_j, se incurrirá en una tardanza T_j, la cual puede calcularse como:

$$T_j = \max(C_j - d_j\ , 0) \qquad \text{Eq.24}$$

- El objetivo consiste en minimizar la tardanza total ΣT_j

3.2. Modelo lineal propuesto.

El modelo lineal propuesto está basado en el concepto de posiciones (*Lasserre & Queyranne 1992),* considerando la existencia de n posiciones para cada

una de las m máquinas, teniendo la posición $l \in L$, donde L es el conjunto de posiciones en el sistema para cada máquina $L = \{1,2 \dots n\}$. Cabe aclarar que es necesario considerar tantas posiciones como tareas, de tal manera que el modelo tenga la posibilidad de asignar todas las tareas a una sola máquina, situación que sería solo favorable en problemas cuyas distribuciones de tiempos de procesamiento para las tareas en las distintas máquinas perteneciera a distintas distribuciones altamente dispares. En estos casos considerar una cantidad reducida de posiciones podría excluir la solución óptima del espacio de búsqueda.

El problema $R\,/d_j/\,\Sigma T_j$ es de alta complejidad *(Du & Leung 1990)*, y anteriormente se ha abordado mediante modelos de programación lineal que requieren de restricciones del tipo gran M, un estudio de la eficiencia de dichos modelos aplicadas a problemas de secuenciación puede encontrarse en el trabajo de *Unlu & Mason (2010)*. El enfoque de posiciones se utilizó anteriormente por *Bruno, Coffman Jr & Sethi (1974)* de manera eficiente para el problema $R\,//\,\Sigma C_j$, en el cual el problema es conceptualizado como un problema de transporte, resolviendo el problema a optimalidad en tiempo polinomial, pero hasta la fecha dicha concepción no ha sido aplicada en el problema $R\,/d_j/\,\Sigma T_j$. Debido a la concepción de posiciones y a que en el sistema actual no se consideran tiempos ociosos, los tiempos de completación de las tareas pueden ser asociados a las posiciones en las que dichas tareas fueron asignadas, lo cual resulta en un cálculo más simple de los tiempos de completación.

Para el modelo en cuestión se considera la variable de decisión binaria Y_{ijl} , que tomará el valor de 1 si la máquina i procesara la tarea j en la posición l, o 0 en caso contrario. También se considerarán las variables continuas C_{il} y T_{il}, que representan el momento de completación y la tardanza asociada a la posición l en la máquina i, respectivamente. El modelo correspondiente se presenta a continuación:

$$\min \sum_{i \in M} \sum_{l \in L} T_{il} \qquad \text{Eq.25}$$

$$\sum_{i \in M} \sum_{l \in L} Y_{ijl} = 1 \qquad \forall\, j \in N \qquad \text{Eq.26}$$

$$\sum_{j \in N} Y_{ijl} \leq 1 \qquad \forall i \in M, l \in L \qquad \text{Eq.27}$$

$$C_{i1} = \sum_{j \in N} p_{ij} Y_{ij1} \qquad \forall\, i \in M \qquad \text{Eq.28}$$

$$C_{il} \geq C_{il-1} + \sum_{j \in N} p_{ij} Y_{ijl} \qquad \forall\, i \in M, l \in L, l > 1 \qquad \text{Eq.29}$$

$$T_{il} \geq C_{il} - \sum_{j \in N} d_j Y_{ijl} \qquad \forall\, i \in M, l \in L \qquad \text{Eq.30}$$

$$C_{il} \geq 0 \qquad \forall\, i \in M, l \in L \qquad \text{Eq.31}$$

$$T_{il} \geq 0 \qquad \forall\, i \in M, l \in L \qquad \text{Eq.32}$$

$$Y_{ijl} \in \{0,1\} \qquad \forall\, i \in M, l \in L \qquad \text{Eq.33}$$

Donde la función objetivo (Eq.25) busca minimizar la tardanza total del sistema. El grupo de restricciones Eq.26 asignan cada tarea a una sola posición en una sola máquina. El grupo de restricciones Eq.27 evita que se asignen múltiples tareas a una sola posición. El grupo de restricciones determinado por las ecuaciones Eq.28 y Eq.29 calculan los tiempos de completación de las distintas posiciones, mientras que las restricciones Eq.30 calculan la tardanza para cada posición de

cada máquina. Por último, las ecuaciones Eq.31, Eq.32 y Eq.33 denotan la naturaleza de las variables.

Este modelo cuenta con mn^2 variables binarias, $2mn$ variables continuas y $3mn + n$ restricciones.

3.2.1. Implicaciones del enfoque de posiciones en la solución factible y ejemplo ilustrativo.

El modelo previamente presentado, debido a su formulación, resulta en soluciones poco intuitivas, las cuales serán descritas y aclaradas en esta sección mediante un ejemplo ilustrativo. Para este ejemplo se considerará un sistema con 6 tareas y 2 máquinas, el cual contará con los siguientes tiempos de procesamiento y fechas de entrega:

$$p_{ij} = \begin{bmatrix} 48 & 30 & 25 & 51 & 36 & 55 \\ 11 & 50 & 15 & 18 & 45 & 32 \end{bmatrix}$$

$$d_j = [80 \quad 70 \quad 75 \quad 91 \quad 96 \quad 90]$$

Se considerará para el ejemplo la siguiente asignación de tareas a las 2 máquinas disponibles, mostrados en las secuencias S_1 y S_2, respectivamente:

$$S_1 = \{6,3\}$$

$$S_2 = \{2,4,1,5\}$$

En esta asignación, se asigna a la máquina uno para trabajar primero la tarea 6 y posteriormente la tarea 3, mientras que la máquina 2 iniciará con la tarea 2, después trabajará la 4, posteriormente la 1, y finalizará con la tarea 5. Dicha secuencia puede ser expresada en el siguiente diagrama de Gantt:

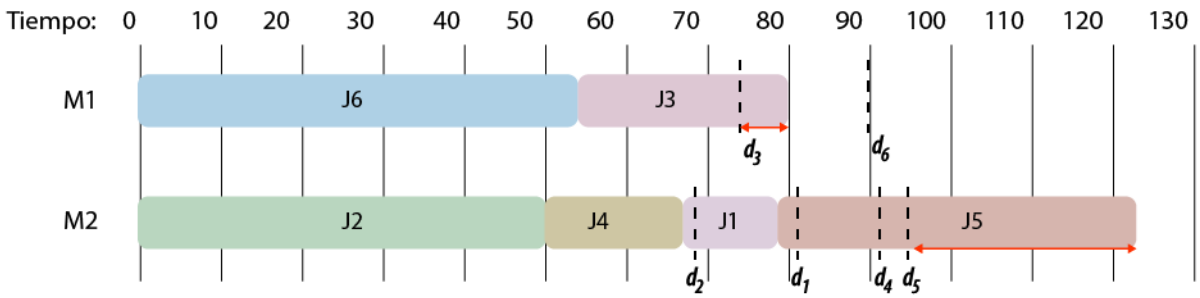


Figura 2: Diagrama de Gant para la secuencia ilustrativa.

En el diagrama mostrado en la Figura 2 se representan cada tarea *j* mediante un bloque distinto, y las fechas de entrega son representadas mediante líneas negras punteadas. En este ejemplo se tendría tardanza de 5 unidades de tiempo para la tarea 3, y 28 unidades de tiempo para la tarea 5 (representado por las líneas rojas). Pensando en la secuencia, y aplicándola al sistema de posiciones, la asignación intuitiva seria la siguiente:

Posición	1	2	3	4	5	6
M1	**6**	**3**	-	-	-	-
M2	**2**	**4**	**1**	**5**	-	-

Figura 3: asignación intuitiva para la secuencia a posiciones

En la secuencia de la Figura 3, las posiciones posteriores quedan sin asignación, quedando las posiciones iniciales con las tareas de la secuencia, las cuales se podrían asociar intuitivamente al orden en que las tareas son trabajadas por sus máquinas correspondientes, siendo la tarea 6 la primera en ser trabajada por la máquina 1, por nombrar un ejemplo. Sin embargo, el modelo propuesto no resultará en dicha asignación, debido a que el grupo de ecuaciones Eq.29 calcula los tiempos de completación asociados a cada posición en base al de la posición anterior. Debido a esto, todas las posiciones vacías (sin tarea asignada), que sean

posteriores a una posición con una tarea asignada, tendrán un tiempo de completación distinto de cero, el cual a su vez resultará en tardanza positiva, de acuerdo con el grupo de ecuaciones Eq.30.

El efecto anterior puede ejemplificarse si se aplican las restricciones Eq.29 y Eq.30 a la posición 3 de la máquina 1 del ejemplo ilustrativo de la asignación mostrada en la Figura 3 , en donde dicha posición no tiene tarea asignada. Simplificando las restricciones anteriores se tendría, $C_{13} \geq C_{12}$ y $T_{13} \geq C_{13}$, forzando erróneamente a que $C_{13} = 80\ y\ T_{13} = 80$. Dicho efecto resultará en una solución indebidamente penalizada en su valor objetivo, llevando al modelo de programación lineal a colocar las tareas al final de la secuencia, para evitar dicho efecto, como se muestra en la Figura 4:

Posición	1	2	3	4	5	6
M1	-	-	-	-	**6**	**3**
M2	-	-	**2**	**4**	**1**	**5**

Figura 4: Representación de la solución para el modelo de programación lineal

Que contará con los tiempos de completación mostrados en la Figura 5:

Posición	1	2	3	4	5	6
M1	0	0	0	0	**55**	**80**
M2	0	0	**50**	**68**	**79**	**124**

Figura 5: Tiempos de completación de la asignación ilustrativa

Y las tardanzas para cada posición mostradas en la Figura 6:

Posición	1	2	3	4	5	6
M1	0	0	0	0	**0**	**5**
M2	0	0	**0**	**0**	**0**	**28**

Figura 6: Tiempos de tardanza de la asignación ilustrativa

Por lo cual, la solución del ejemplo tendrá un valor de tardanza total de 33 unidades de tiempo.

3.2.2. Soluciones equivalentes y desigualdades válidas.

Debido a la forma en que se crea la solución, se puede notar que existen soluciones equivalentes subóptimas para cualquier secuencia, siendo éstas las que se produzcan por cualquier acomodo de tareas en posiciones que no sean las últimas. Un ejemplo se muestra en la siguiente Figura 7:

Posición	1	2	3	4	5	6
M1	-	-	-	**6**	-	**3**
M2	-	-	**2**	**4**	**1**	**5**

Figura 7: Representación de una solución subóptima para el ejemplo ilustrativo

Esta solución es equivalente a la mostrada en la Figura 4, sin embargo, al no tener todas las tareas asignadas a sus últimas posiciones disponibles (como se resalta en la posición 5 de la máquina 1), se contará con tiempos de completación diferentes según se muestra en la siguiente Figura 8:

Posición	1	2	3	4	5	6
M1	0	0	0	**55**	55	**80**
M2	0	0	**50**	**68**	**79**	**124**

Figura 8: Tiempos de completación de la asignación subóptima

Es notorio que en la posición 5 de la de la máquina 1, a pesar de no haber ninguna tarea asignada, el tiempo de completación es 55 (resaltado en rojo), debido

a la recursividad del grupo de ecuaciones Eq.29. Asimismo, al no haber una tarea asignada, el grupo de ecuaciones Eq.30 no podrá recuperar ningún valor de fecha de entrega para compararlo con el momento de completación, resultando en una tardanza de 55 para dicha posición, teniendo así una tardanza total de 88 unidades de tiempo (Figura 9).

Posición	1	2	3	4	5	6
M1	0	0	0	**0**	55	**5**
M2	0	0	**0**	**0**	**0**	**28**

Figura 9: Tiempos de tardanza de la asignación de la asignación subóptima

Las soluciones anteriormente mencionadas hacen más extensa la cantidad de soluciones en la región factible; lo que resulta en un proceso de exploración más lento. Para hacer más eficiente el proceso de exploración de la región factible, se propone el grupo de restricciones Eq.34, que descarta aquellas soluciones cuyas asignaciones no estén asociadas a las últimas posiciones disponibles, reduciendo así la cantidad de soluciones por ser exploradas:

$$\sum_{j \in N} Y_{ijl-1} \leq \sum_{j \in N} Y_{ijl} \qquad \forall\, i \in M, l \in L \qquad \text{Eq.34}$$

Como se mostrará en la sección (3.4. Experimentación), la inclusión de dichas desigualdades validas acelera considerablemente el proceso de tiempo requerido para obtener la solución del modelo propuesto.

3.3. Algoritmo metaheurístico.

Como alternativa al uso del modelo de programación lineal, se propone el algoritmo metaheurístico de búsqueda local iterada, el cual será descrito en esta sección. Como se demostrará en la sección (3.4. Experimentación), el algoritmo

metaheurístico propuesto puede abordar instancias de mayor tamaño que el modelo de programación lineal propuesto.

En esta sección se describirá el algoritmo en general, los distintos procedimientos que lo componen, y se discutirán metodologías para la implementación que hagan más eficiente la ejecución de tales procedimientos.

3.3.1. Representación de la solución.

Para el algoritmo propuesto se utilizará la representación de la solución intuitiva, similar a la mostrada en la Figura 3, donde el número de la posición asignada a las tareas se podrá asociar directamente al orden en que dichas tareas serán procesadas. Esta información será guardada en una matriz cuyas dimensiones estarán asociadas a las posiciones que existen en las distintas máquinas, almacenando dentro de tal matriz el número de la tarea asignada a dicha posición. Esto se explicará más a detalle en la siguiente sección.

Para la asignación del ejemplo de la Figura 3 se contaría entonces con la siguiente matriz:

$$\begin{bmatrix} 6 & 5 & 0 & 0 & 0 & 0 \\ 2 & 4 & 1 & 5 & 0 & 0 \end{bmatrix}$$

En dicha matriz se pueden considerar los valores de cero como una posición sin tarea asignada. Cabe aclarar que, aunque se define la representación como una matriz, dado que no se harán cálculos con la matriz completa en ningún procedimiento del algoritmo, se podría utilizar en su lugar múltiples vectores durante su implementación.

3.3.2. Estructura general del Algoritmo propuesto.

Esta sección describe el algoritmo de búsqueda local iterada propuesto para resolver el problema $R\ /\ d_j\ /\ \Sigma T_j$. El algoritmo consta de tres procedimientos principales: (1) un procedimiento de construcción inicial, (2) un procedimiento de mejora mediante búsqueda local y (3) un procedimiento de perturbación para diversificar el área de búsqueda.

```
   //Algoritmo de búsqueda local iterada
   Datos: m, n, D, P, U
 1 iter = 1
 2 (A, R, T) ← Proceso Constructivo (m, n, D, P, U)
 3 (A*, R*, T*) ← Fase 1 del proceso de Mejora (A, R, T)
 4 (A*, R*, T*) ← Fase 2 del proceso de Mejora (A*, R*, T*)
 5 Mientras iter ≤ Maxiter hacer:
 6     (A, R, T) ← Proceso de Perturbación (A*, R*, T*)
 7     (A', R', T') ← Fase 1 del proceso de Mejora (A, R, T)
 8     (A', R', T') ← Fase 2 del proceso de Mejora (A', R', T')
 9     Si T' < T* entonces:
10         (A*, R*, T*) ← (A', R', T')
11         iter = 0
12     Fin
13     iter = iter + 1
14 Fin
   Resultado: A*, T*
```

Figura 10. Pseudocódigo General del Algoritmo

En la Figura 10 se muestra el pseudocódigo para estructura general del algoritmo, en el cual se consideran los siguientes definiciones: El algoritmo requiere como entradas el número de trabajos n, el número de máquinas m, un vector D de dimensión n, en el cual cualquier elemento d_j representa la fecha de vencimiento del trabajo j, además de una matriz P de dimensión $m \times n$, en la que cualquier

elemento p_{ij} denota el tiempo que requiere el trabajo j para ser procesado en la máquina i. Finalmente, se necesita una lista de tareas a secuenciar U.

Se representará la solución factible dada por el algoritmo mediante una matriz A de dimensión $m \times n$, en la que cualquier elemento a_{il} almacenará el índice del trabajo asignado en la posición l de la máquina i. Adicionalmente se necesitará el vector R de dimensión m, en la que cualquier elemento r_i denotará el número de trabajos asignados a la máquina i, y el parámetro escalar T, que representa la tardanza total de una determinada secuencia de trabajos. De manera similar a la formulación matemática, el algoritmo considera $m \times n$ posiciones, en las que se asignarán los n trabajos, y las demás posiciones permanecerán vacías.

Como se muestra en la línea 2 de la Figura 10, el procedimiento constructivo recibe como entrada los parámetros $m, n, D\ y\ P$ para crear una solución inicial con la siguiente estructura: una matriz de asignación A, un vector R en el que cada elemento denota el número de trabajos por máquina y un parámetro T que denota la tardanza total de la solución. Una vez construida, la solución inicial pasa a la primera fase del procedimiento de mejora, buscando reasignar los trabajos de manera que se pueda obtener una solución con un menor valor de tardanza. Después de esto, la solución resultante pasa a la segunda fase de mejora, buscando una nueva mejora a través de un intercambio de trabajos entre máquinas.

Para diversificar el espacio de búsqueda, se aplica un procedimiento de perturbación sobre la mejor solución encontrada hasta el momento (A^*, R^*, T^*) en busca de encontrar un nuevo mínimo local (A', R', T'); si la solución obtenida en cada iteración de perturbación-mejora tiene un menor valor de tardanza que la mejor solución obtenida hasta el momento, entonces se actualiza la mejor solución actual. Vale la pena señalar que el ciclo de perturbación-mejora se aplicará hasta que se haya ejecutado un determinado número de iteraciones consecutivas (denotado por el parámetro $Maxiter$) sin encontrar una mejora.

3.3.3. Procedimiento constructivo.

El procedimiento constructivo genera una solución inicial en la que todos los trabajos son asignados y secuenciados en las máquinas para que la tardanza total sea la mejor posible. El mecanismo de asignación funciona de la siguiente manera: durante cada iteración, se selecciona secuencialmente (por orden) un trabajo de la lista U de trabajos no asignados, el trabajo seleccionado se prueba en todas las posiciones existentes para cada máquina en el sistema, verificando el correspondiente incremento en la tardanza (ΔT) del sistema, y luego se asigna a su mejor posición (en caso de empate, se conserva la primera mejor posición encontrada). En otras palabras, cada trabajo se evalúa sobre cada una de las posibles $ri + 1$ posiciones en la máquina i. Este procedimiento se repite para cada máquina. En el caso de tener una máquina vacía, solo se evalúa la primera posición. En caso de que una máquina tenga previamente asignado un trabajo, las posibles posiciones a evaluar serán 2, y así sucesivamente. Posteriormente, el procedimiento asigna el trabajo a la máquina y posición que produce el mínimo valor de incremento en la tardanza ΔT (mejor posición). El procedimiento finaliza cuando todos los trabajos han sido asignados, es decir, cuando la lista U está vacía. La lista U se puede crear según diferentes criterios. En este caso, se han considerado cuatro criterios: aleatoria (RND), fecha de vencimiento más temprana (EDD), índice de prioridad de tráfico (TPI) y holgura mínima (MSL). La Figura 11 muestra el pseudocódigo de este método constructivo.

```
   //Proceso Constructivo
   Datos: m, D, P, U, A
1  Inicializar A = 0, R = 0, T = 0
2  Ordenar U mediante uno de los métodos (RND, EDD, TPI, MSL)
3  Hacer:
4      Seleccionar el siguiente trabajo j de U
5      Para cada máquina i hacer:
```

6 **Para cada posición** k **entre** 1 **y** $ri + 1$ **hacer:**
7 Calcular ΔT por asignar el trabajo j en la máquina i en la posición k
8 **Fin**
9 **Fin**
10 Asignar trabajo j en la máquina i en la posición k que produzca el menor ΔT
11 Eliminar trabajo j de U
12 **Mientras** $U \neq \emptyset$
13 Calcular T
Resultado: A, R, T

Figura 11: Pseudocódigo del Proceso Constructivo

La descripción de los métodos de clasificación propuestos para ordenar U se proporciona a continuación:

- RND: los trabajos se ordenan aleatoriamente
- EDD: Los trabajos se ordenarán de forma no decreciente por sus respectivos valores de fecha de vencimiento. Es decir, los trabajos que deben entregarse antes deben secuenciarse primero.
- TPI: Introducido por *Ho & Chang (1991)* para el problema de máquinas paralelas idénticas, TPI es un índice que mide qué tan congestionado está un sistema. Este índice combina información de las reglas de despacho EDD y SPT. En esta tesis, se implementaron pocos cambios para adaptarse al sistema actual. Los cálculos con respecto a este procedimiento son los siguientes: Se determinará un índice de congestión de tráfico (TCR) por $TCR = (\hat{p}n)/(\hat{d}m)$, donde $\hat{p}$ y $\hat{d}$ son los promedios de todos los tiempos de procesamiento y fechas de entrega, respectivamente. Sean W_d y W_p pesos asociados a las reglas de despacho EDD y SPT, respectivamente. Se calculan como $W_d = max(min(0.5 + (3 - TCR)/TCR, 1), 0)$ y $W_p = 1 - W_d$. Se calculará un índice IN_j para cada trabajo j mediante la expresión $IN_j = (W_d d_j/maxd) + (W_p \max_i(p_{ij})/maxp)$, donde $maxd$ y $maxp$ serán la fecha de entrega más grande y el tiempo de procesamiento más largo,

respectivamente. Los trabajos estarán ordenados de manera ascendente según su índice IN_j.

- MSL: Todos los trabajos serán ordenados en forma ascendente por sus respectivas holguras SL_j, que se calculan para cada tarea j mediante la expresión $SL_j = d_j - \max_i(pij)$.

Como se verá más adelante, la calidad de la solución inicial depende principalmente del orden en que se asignaron los trabajos. Esto ocurre porque, de manera secuencial, el siguiente trabajo a asignar se evalúa en la solución parcial obtenida hasta el momento.

Dado que se tienen cuatro estrategias de ordenamiento diferentes para crear la lista U, se implementaron cuatro procedimientos constructivos diferentes y, por lo tanto, se diseñaron cuatro versiones diferentes del algoritmo obtenido. Para la siguiente etapa del algoritmo, 2 fases de mejora fueron incorporadas, las cuales son descritas a continuación.

3.3.4. Procedimiento de mejora.

El procedimiento de mejora se compone de 2 fases. La primera consiste en un mecanismo de reasignación exhaustivo para mejorar la tardanza total obtenida con la solución inicial. En esta primera fase se construye una lista de trabajos, donde se ordenan los trabajos en orden descendente por su valor de tardanza (en el caso de tener empates, los trabajos se ordenan en orden lexicográfico). Se selecciona el primer trabajo de la lista. Posteriormente, se extrae de su posición en la solución actual y se prueba en todas las posiciones posibles de cada máquina en la solución (incluida su máquina actual), para encontrar una mejor posición (una ubicación que disminuya el valor de tardanza total actual).

Si se encuentra una mejor posición, el trabajo se reasigna, la lista se actualiza y el proceso comienza de nuevo. De lo contrario, el trabajo seleccionado permanece en su posición original y se evalúa el siguiente trabajo de la lista. La primera fase del procedimiento finaliza hasta que se han analizado todos los trabajos de la lista sin encontrar ninguna mejora.

La Figura 12 muestra el pseudocódigo de esta primera fase del procedimiento de mejora.

```
   //Fase 1 del proceso de Mejora
   Datos: A, R, T
1  O = vector que contiene todas las tareas
2  Registrar la tardanza total (T) de la solución A
3  Ordenar el vector O descendente según la tardanza de cada tarea
4  Para cada trabajo j ∈ O hacer:
5      Extraer trabajo j de A
6      Para cada máquina i hacer:
7          Determine la posición con el valor mínimo de tardanza en la máquina i
8      Fin
9      Determine la mejor posición k en la máquina i' con el menor valor de tardanza
       y calcular el valor de tardanza total temporal T'
10     Si T' < T entonces:
11         Actualizar A asignando el trabajo j la mejor posición k en la máquina i'
12         Recalcular O
13         Ir a la línea 2
14     En caso contrario:
15         Reasignar j a su posición original
16     Fin
17 Fin
   Resultado: A, R, T
```

Figura 12: Pseudocódigo de la Fase 1 del proceso de Mejora

Como se puede observar, la fase 1 realiza una búsqueda exhaustiva en el vecindario de la solución actual utilizando la mejor posición de reubicación para primer trabajo que logre una mejora (pasos 4 a 14).

La segunda fase evaluará el intercambio de trabajos entre dos máquinas, conservando únicamente los movimientos que mejoran la solución: estos intercambios se evalúan de forma extensiva, probando para cada máquina, intercambiando cada uno de los trabajos de cada puesto con cada uno de los trabajos de cada puesto de las otras máquinas. Si se encuentra un intercambio que vale la pena, el algoritmo ejecuta el movimiento y continúa la búsqueda desde la máquina y la posición actuales. La búsqueda se detiene después de evaluar todas las máquinas y posiciones disponibles. Un mecanismo adicional que ayuda a diversificar el espacio de búsqueda fue diseñado, el cual se describe a continuación.

La segunda fase del algoritmo se muestra a continuación (Figura 13).

```
   //Fase 2 del proceso de Mejora
   Datos: A, R, T
 1  Calcular la tardanza total (T) de la solución A
 2  Para cada máquina i < m hacer:
 3      Para cada tarea j en la máquina i hacer:
 4          Para cada máquina i' > i hacer:
 5              Para cada trabajo j' en la máquina i' hacer:
 6                  A' ← A
 7                  Cambiar el trabajo j de la máquina i con el trabajo j' de la máquina i' en A
 8                  Calcular la tardanza T' de la solución A'
 9                  Si T' < T entonces:
10                      (A, T) ← (A', T')
11                  Fin
12              Fin
13          Fin
14      Fin
15  Fin
   Resultado: A, R, T
```

Figura 13: Pseudocódigo de la Fase 2 del proceso de Mejora

3.3.5. Proceso de perturbación.

Este procedimiento tiene como objetivo generar distintas soluciones iniciales para el procedimiento de mejora. Consiste en reubicar aleatoriamente trabajos desde su posición actual a una posición diferente, ya sea en la misma máquina o en otra nueva. Este procedimiento se puede ejecutar varias veces con el propósito de ampliar el área de búsqueda. Sin embargo, el porcentaje de trabajos reubicados será siempre el mismo y estará definido por el parámetro njr. La mejor solución obtenida hasta el momento será siempre seleccionada para la perturbación. La Figura 14 muestra el pseudocódigo de este procedimiento.

```
   //Proceso de perturbación
   Datos: A, R, T
1  Hacer:
2      Seleccionar arbitrariamente una tarea j de la solución A
3      Reasignar arbitrariamente la tarea j en una posición diferente en A
4  Mientras njr tareas han sido reasignadas
5  Actualizar el valor de la tardanza total (T)
   Resultado: A, R, T
```

Figura 14: Proceso de perturbación

3.3.6. Estrategia de aceleración.

Tanto los procedimientos constructivos como los de mejora exigen extensos cómputos durante la búsqueda de una posición y máquina en la cual insertar un trabajo. Adicionalmente, cualquier trabajo en una posición posterior a la insertada experimentará un cambio en su tiempo de finalización y, por lo tanto, su tardanza podrá cambiar. Un enfoque intuitivo para realizar tales cálculos puede consistir en agregar el trabajo en la primera posición, calcular los cambios para todas las posiciones posteriores, extraer el trabajo y repetir este procedimiento para todas las

demás posiciones de la máquina, la cual implica la actualización de una gran cantidad de valores. A continuación, se presenta un enfoque mejorado para realizar un cálculo de este tipo que reduce la cantidad de datos que se necesitan actualizar, el cual está inspirado en el algoritmo de ordenamiento de burbujas que se muestra en el trabajo de *Ho & Chang (1991)*.

La metodología propuesta parte de la consideración que, para cualquier secuencia dada, si dos trabajos adyacentes en la secuencia cambian de lugar, todos los trabajos que les preceden no percibirán ningún cambio. El procedimiento propuesto consiste en colocar el trabajo a reasignar al final de la secuencia, y luego cambiar posiciones entre trabajos adyacentes, hasta que el trabajo se desplace hasta el principio de la secuencia. Durante cada movimiento, solo los trabajos que cambian de lugar necesitan actualizar su información.

Considere la segunda máquina en el ejemplo presentado en la Figura 3. Suponga que, durante la fase de mejora, el trabajo 4 se extrajo de su posición actual (segunda posición) en la máquina 2 y se probará en las demás posiciones relevantes de dicha máquina para determinar si insertarlo en una posición diferente puede traer una mejora, como se muestra en la Figura 15 (por razones prácticas, solo se considerarán cuatro posiciones en la ilustración).

Posición	1	2	3	4
Secuencia	2	1	5	4
Completación	50	61	106	124
Tardanza	0	0	10	33

(a) tarea 4 insertada al final

Posición	1	2	3	4
Secuencia	2	1	4	5
Completación	50	61	79	124
Tardanza	0	0	0	28

(b) tarea 4 y 5 intercambiadas

Posición	1	2	3	4
Secuencia	2	4	1	5
Completación	50	68	79	124
Tardanza	0	0	0	28

(c) tarea 4 y 1 intercambiadas

Posición	1	2	3	4
Secuencia	4	2	1	5
Completación	18	68	79	124
Tardanza	0	0	0	28

(d) tarea 4 y 2 intercambiadas

Figura 15: Proceso de Aceleración de inserción

Como se muestra en la Figura 15, el trabajo 4 se inserta al final de la secuencia (Figura 15a) y se cambia hacia el comienzo de la secuencia (Figura 15b, Figura 15c, Figura 15d)), actualizando solo los valores sombreados en gris durante cada cambio.

3.4. Experimentación.

En esta sección se describe todo lo concerniente a la experimentación llevada a cabo. Específicamente, se describirán las metodologías para la creación de las instancias, los experimentos realizados, y los resultados obtenidos.

3.4.1. Metodología y creación de instancias.

Para evaluar el desempeño del modelo y el algoritmo metaheurístico, se crearon dos conjuntos diferentes de instancias usando tiempos de procesamiento p_{ij}, generados considerando una distribución uniforme discreta, $p_{ij} \sim U[1, 99]$ para el primer conjunto y $p_{ij} \sim U[51, 99]$ para el segundo conjunto; En adelante usaremos la notación $U[a, b]$ cuando una variable x se generó a partir de una distribución uniforme discreta en el rango entre a y b, es decir, si $x \sim [a, b]$.

Estas distribuciones y rangos corresponden a los comúnmente usados por *Potts & Van Wassenhove (1982)*, en *Shim & Kim (2007)b*, y en Shim & Kim (2007) para problemas de programación de máquina única, máquina paralela idéntica y máquina paralela no relacionada, respectivamente; y por *Vallada, Ruiz & Minella(2008)* para problemas de programación de máquinas paralelas no relacionados con tiempos de preparación.

Con respecto a la cantidad de trabajos y máquinas, se consideraron varias combinaciones con el fin de crear problemas de tamaño pequeño, mediano y

grande. El grupo de instancias pequeñas está formado por las posibles combinaciones de valores para $n \in \{14,16,18,20\}$ y $m \in \{2,3,4\}$. Para las instancias de tamaño mediano $n \in \{50,100,150\}$ y $m \in \{5,10,15,20\}$. Finalmente, para las instancias de tamaño grande, los valores para cantidad de máquinas y tareas son $n \in \{200,300,400\}$ y $m \in \{5,10,15,20\}$, respectivamente. Las fechas de vencimiento d_j se generaron usando una distribución uniforme discreta como $d_j \sim U[\bar{p}(1-\alpha-\beta/2), \bar{p}(1-\alpha+\beta/2)]$, donde α es un parámetro de estrechez y β un parámetro de rango, donde ambos pueden tomar cualquier valor entre 0 y 1. Con respecto al parámetro de estrechez α y el parámetro de rango β, se definieron 3 niveles para cada uno: $\alpha \in \{0.2,0.4,0.6\}$, y $\beta \in \{0.2,0.5,0.8\}$. El valor de $\bar{p}$ se discutirá en detalle a continuación.

Históricamente, el parámetro $\bar{p}$ se ha calculado de diferentes maneras. *Potts & Van Wassenhove (1982)* consideró $\bar{p}$ como la duración óptima para el problema de secuenciación de una sola máquina, dado por $\bar{p} = \sum_j P_j$.En el trabajo de *Shim & Kim (2007)b se* consideró una aproximación al makespan (C_{max}) dado por $\bar{p} = \sum_j P_j / m$ para el problema de secuenciación en máquinas paralelas. Los mismos autores propusieron un makespan aproximado $\bar{p} = \sum_j \sum_i P_{ij} / m^2$ para el problema de secuenciación en máquinas paralelas no relacionado *(Shim & Kim 2007)*.

La Figura 16 muestra una representación de cómo el valor de $\bar{p}$ se aproxima a C_{max} (el intervalo de producción óptimo), lo que resulta en un cambio en el rango de generación para las fechas de vencimiento (considerando los rangos A y B). Es notable que cuando se selecciona un valor grande de α, el rango se generará cerca de la izquierda y el rango se generará más amplio cuando se seleccionen valores más grandes de β.

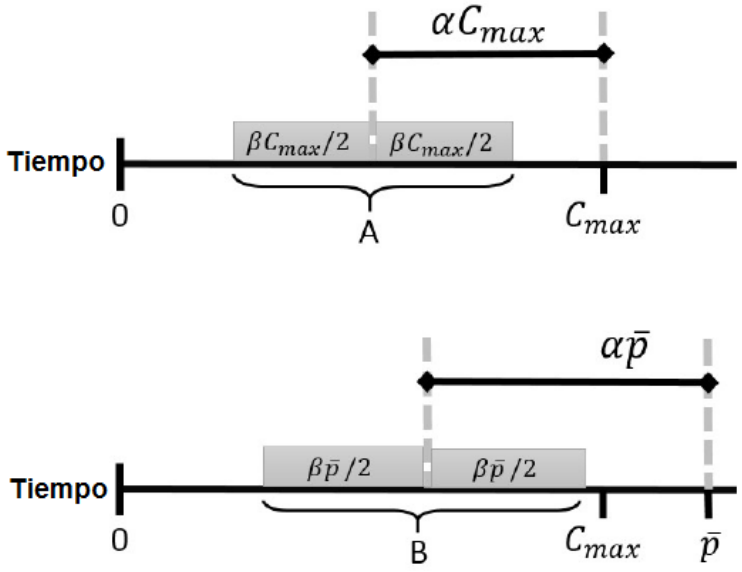


Figura 16: Rango de generación de d_j para valor de $\bar{p}$ en comparación con C_{max}

Este método de construcción de la instancia se ha utilizado anteriormente en la literatura *(Shim & Kim 2007),* pero puede producir instancias con una holgura considerable debido al procedimiento de aproximación de $\bar{p}$, que genera grandes valores de fechas de entrega.

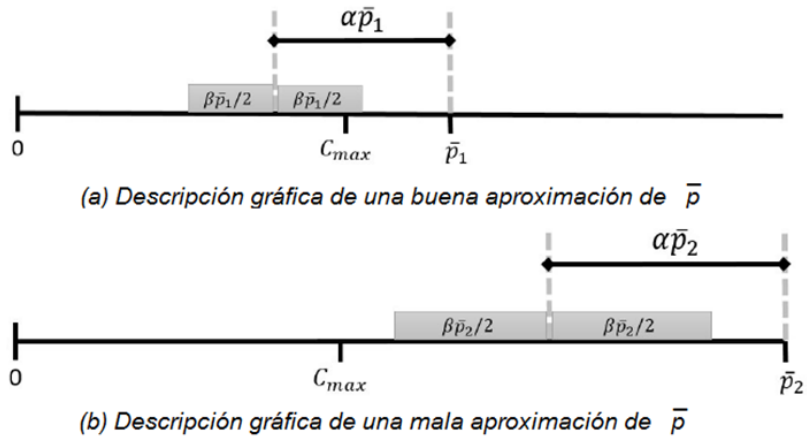


Figura 17: Efectos de la aproximación de makespan en la generación del rango de fechas de vencimiento para los mismos valores de α y β

La Figura 17 muestra cómo una aproximación incorrecta de $\bar{p}$ afecta negativamente el rango de generación de fechas de vencimiento. Específicamente, la Figura 17a ilustra cómo un rango adecuado creará fechas de vencimiento razonables, mientras que la Figura 17b crea fechas de vencimiento que difícilmente se infringirán (puesto que son posteriores al Makespan). Entonces, podemos observar que a medida que el valor de $\bar{p}$ se acerca al valor de C_{max}, las fechas de vencimiento generadas son más realistas. Por lo tanto, proponemos considerar $\bar{p} = C_{max}$.

Se realizó una experimentación preliminar para comparar la complejidad (en términos del tiempo computacional transcurrido) de las instancias creadas por ambos $\bar{p} = \sum_j \sum_i P_{ij} /m^2$ y $\bar{p} = C_{max}$. En la Tabla 2 se presenta, de manera resumida, por tiempos de procesamiento (TiempoProc) y número de trabajos (n), el número de instancias probadas (Inst), el número de aquellas instancias que tuvieron un valor de función objetivo de cero (OF=0) y el tiempo medio transcurrido (Tiempo) para $\bar{p}$, etiquetados como $\overline{p_1}$ y $\overline{p_2}$ donde $\overline{p_1} = \sum_j \sum_i P_{ij} /m^2$ y $\overline{p_2} = C_{max}$.

Tabla 2: comparación de complejidad entre $\overline{p_1}$ y $\overline{p_2}$

		$\overline{p_1}$			$\overline{p_2}$		
TiempoProc	n	Inst	OF=0	Tiempo	Inst	OF=0	Tiempo
[1,99]	14	135	89	1.2	135	8	0.5
	16	135	90	1.4	135	10	0.5
	18	135	96	3.1	135	11	2.9
	20	135	93	73.1	135	12	2.2
	50	180	162	258.7	180	9	1,042.9
	100	180	161	658.4	180	17	4,398.6
	150	180	162	760.2	180	22	5,383.5
[51,99]	14	135	40	0.3	135	17	0.2
	16	135	41	0.8	135	19	0.3
	18	135	48	0.4	135	17	0.5
	20	135	49	2.6	135	18	1.1
	50	180	82	531.3	180	24	526.7
	100	180	100	1,671.0	180	37	1,273.3
	150	180	100	2,741.1	180	40	2,851.9
Total		2160	1313	556.9	2160	261	1,290.3

Se puede notar que para instancias creadas con $\overline{p_1} = \sum_j \sum_i P_{ij} \, / m^2$ hasta el 60.8% de ellas reportaron un valor de la función objetivo de 0, en comparación con las instancias creadas con $\overline{p_2} = C_{max}$, donde solo el 12.1% de ellas reportaron un valor objetivo de 0, lo que confirma que el procedimiento propuesto para la generación de instancias produce fechas de vencimiento más estrictas.

Debido a esto, se usará $\overline{p} = C_{max}$como estimador para la creación de las instancias finales en este trabajo. El modelo lineal utilizado para calcular C_{max} se muestra en el Apéndice B.

En total, se considerarán 648 combinaciones posibles $\{p, \alpha, \beta, n, m\}$; Para cada combinación, se generaron 5 instancias, dando un total de 3240 instancias finales.

3.4.2. Resultados de la formulación matemática y comparativa con modelos de la literatura.

Para las formulaciones lineales, éstas se codificaron utilizando el lenguaje AMPL y se resolvieron utilizando GUROBI 9.1.0, estableciendo un tiempo máximo de ejecución de 7200 segundos (2 horas). Los experimentos se realizaron en una PC con procesador Intel Xeon e5-1650 v2 @ 3.5 GHz y 8 GB de RAM, utilizando el sistema operativo Windows 10.

Para evaluar el desempeño de las formulaciones, se utilizó primero un subconjunto de 270 instancias (todas con tamaño de $n = 14$ trabajos) con el propósito de validar el modelo propuesto contra el modelo más afín adaptado en la literatura. El modelo propuesto utilizado para dicha comparación es la formulación original, que comprende de la Eq.25 a Eq.33, al que se denominará MILP1, mientras que el modelo adaptado de la literatura (*De CM Nogueira, et al. 2014*) será referido como MILP0 (ver Apéndice A).

En la Tabla 3 se pueden ver los resultados de la ejecución de las instancias en el modelo MILP0 y MILP1, agrupados tomando como base los parámetros α y β, además de los tiempos de procesamiento correspondientes $U[1,99]$ o $U[51,99]$. Cada grupo consta de 15 instancias para los diferentes valores de m. En la tabla se muestran las tardanzas totales de los modelos MILP0 y MILP1 (columna TarTot), tiempo de cómputo transcurrido en segundos (columna Tiempo), número de instancias resueltas (columna #Res) y Gap promedio (columna %Gap). El Gap promedio se calculó mediante la expresión(Eq.35):

$$\%Gap = 100 * (VO - CI)/VO \qquad \text{Eq.35}$$

Donde VO y CI representan el valor objetivo y la cota inferior reportada por Gurobi, respectivamente.

Tabla 3: Comparación de resultados entre MILP0 y MILP1

			MILP0				MILP1			
TiempoProc	α	β	TarTot	Tiempo	#Res	%Gap	TarTot	Tiempo	#Res	%Gap
[1,99]	0.2	0.2	40.80	5,523.84	15	0.00	40.80	0.33	15	0.00
		0.5	19.27	3,556.44	14	71.43	19.27	0.41	15	0.00
		0.8	16.80	1,530.71	15	0.00	16.80	1.98	15	0.00
	0.4	0.2	127.40	6,276.69	15	0.00	127.40	0.10	15	0.00
		0.5	116.00	5,401.64	15	0.00	116.00	0.25	15	0.00
		0.8	89.13	4,429.60	15	0.00	89.13	0.43	15	0.00
	0.6	0.2	257.33	6,140.48	15	0.00	257.00	0.09	15	0.00
		0.5	316.40	5,572.58	12	85.36	316.40	0.34	15	0.00
		0.8	260.07	4,657.13	12	67.49	260.07	0.36	15	0.00
[51,99]	0.2	0.2	91.46	7,203.35	15	0.00	91.47	0.16	15	0.00
		0.5	27.66	5,766.40	15	0.00	27.67	0.55	15	0.00
		0.8	2.67	483.78	15	0.00	2.67	0.33	15	0.00
	0.4	0.2	393.73	7,213.92	14	100.00	393.07	0.11	15	0.00
		0.5	250.39	7,211.44	14	100.00	248.67	0.18	15	0.00
		0.8	182.13	6,821.14	14	73.86	179.20	0.17	15	0.00
	0.6	0.2	916.00	7,238.40	15	0.00	915.73	0.09	15	0.00
		0.5	819.27	7,277.52	12	76.74	819.07	0.12	15	0.00
		0.8	925.20	7,237.44	9	68.56	925.20	0.15	15	0.00
Promedio			269.54	5,530.14	251	74.19	269.20	0.34	270	0.00

El resto de la experimentación fue evaluada sobre la totalidad de instancias pequeñas y medianas. Esta experimentación se realizó considerando el modelo MILP1 y comparando su desempeño con el modelo que incorpora las Eq.34, refiriéndose a este modelo extendido como MILP2. En estos experimentos se reporta el número de instancias resueltas a optimalidad dentro del límite de tiempo o el Gap reportado al alcanzarlo. Para las instancias que alcanzaron el límite de tiempo, la brecha ($RelGap$) entre el valor objetivo y la cota inferior se calculó como $RelGap = VO - CI$. Para las instancias resueltas a optimalidad dentro del límite de tiempo, el valor de $RelGap$ se reporta como cero.

La Tabla 4 reporta los resultados sobre el conjunto de instancias pequeñas, agrupando los resultados de acuerdo con sus parámetros α y β, y sus correspondientes tiempos de procesamiento $U[1,99]$ o $U[\,51,99]$. Cada grupo en esta tabla contiene un total de 60 instancias, correspondientes a cinco instancias para cada combinación de n y m. Como todas las instancias del grupo pequeño se resolvieron dentro del límite de tiempo para ambos modelos (MILP1 y MILP2), solo se muestran dos métricas para cada grupo: el valor promedio de la función objetivo (columna TarTot) y el tiempo promedio transcurrido en segundos (columna Tiempo). Adicionalmente, puede notarse como al aumentar el parámetro α aumenta la tardanza promedio debido a que la media de la distribución utilizada para generar las tardanzas estará más cercana al 0, mientras que aumentar el parámetro β genera el efecto contrario debido a que la distribución será más amplia, y al ser uniforme generara un set de fechas de entrega ampliamente distribuidas que será fácil de cumplir; con excepción de la combinación de α=0.6 y β=0.8, el cual resulta en valores altos de tardanza promedio debido al amplio rango en el cual pueden distribuirse las fechas de entrega de las tareas, resultando en varias fechas de entrega estrictas.

Tabla 4: Resultados de las formulaciones para instancias pequeñas

		[1,99]				[51,99]			
		MILP1		MILP2		MILP1		MILP2	
α	β	TarTot	Tiempo	TarTot	Tiempo	TarTot	Tiempo	TarTot	Tiempo
0.2	0.2	45.3	0.3	45.3	0.4	116.2	0.2	116.2	0.2
	0.5	16.4	5.9	16.4	2.0	20.9	1.7	20.9	0.5
	0.8	10.0	1.8	10.0	1.5	3.5	1.1	3.5	0.6
0.4	0.2	170.3	0.2	170.3	0.3	541.2	0.2	541.2	0.2
	0.5	134.4	0.6	134.4	0.7	348.4	0.3	348.4	0.3
	0.8	107.1	3.0	107.1	3.3	189.4	0.7	189.4	0.6
0.6	0.2	372.6	0.2	372.6	0.2	1,268.4	0.1	1,268.4	0.1
	0.5	400.2	0.6	400.2	0.6	1,134.6	0.2	1,134.6	0.2
	0.8	348.8	1.3	348.8	1.5	1,132.0	0.4	1,132.0	0.4
Promedio		178.4	1.6	178.4	1.2	528.3	0.5	528.3	0.4

Como se observa en la Tabla 4, los tiempos promedio de ejecución siguen siendo relativamente cortos, siendo MILP2 ligeramente más rápido que MILP1.

Los resultados para las instancias de tamaño mediano se agruparon de manera similar a la Tabla 4, pero incorporando columnas que indican el porcentaje de instancias resueltas de forma óptima (columna %Res), que se calcularon sobre todos los valores de m, la brecha relativa promedio (columna RelGap), la brecha porcentual promedio (columna %Gap), calculada sobre las instancias en las que el modelo alcanzó el límite de tiempo. Para facilitar la legibilidad, se creó una tabla para cada valor de n, siendo estas la Tabla 5, Tabla 6 y Tabla 7 (para 50, 100 y 150 trabajos, respectivamente). Cada tabla informa los resultados de más de 20 instancias (correspondientes a cinco instancias para todos los valores posibles de m).

Tabla 5: Resultados de la formulación para instancias de 50 trabajos

			MILP1					MILP2				
TiempoProc	α	β	TarTot	%Res	RelGap	%Gap	Tiempo	TarTot	%Res	RelGap	%Gap	Tiempo
[1,99]	0.2	0.2	39.5	100.0	0.0	0.0	80.1	39.5	100.0	0.0	0.0	36.6
		0.5	11.0	80.0	4.0	17.0	2,012.5	11.0	90.0	1.3	4.9	1,239.6
		0.8	9.5	85.0	7.0	35.6	1,575.7	9.5	95.0	4.0	22.2	1,275.7
	0.4	0.2	146.9	100.0	0.0	0.0	49.3	146.9	100.0	0.0	0.0	28.9
		0.5	113.1	95.0	6.0	2.0	723.5	113.1	100.0	0.0	0.0	244.8
		0.8	80.4	75.0	26.2	21.9	2,454.0	80.4	85.0	20.4	31.8	1,826.9
	0.6	0.2	365.2	100.0	0.0	0.0	6.8	365.2	100.0	0.0	0.0	7.6
		0.5	310.9	90.0	11.3	1.6	861.2	310.9	95.0	11.7	1.8	823.5
		0.8	322.1	95.0	64.0	9.3	1,622.7	322.1	90.0	40.8	6.7	1,600.9
[51,99]	0.2	0.2	227.1	100.0	0.0	0.0	77.9	227.1	100.0	0.0	0.0	74.4
		0.5	6.0	50.0	6.0	87.0	3,759.2	6.0	95.0	6.0	100.0	584.5
		0.8	0.2	95.0	3.0	100.0	369.0	0.2	95.0	3.0	100.0	366.6
	0.4	0.2	1,146.7	100.0	0.0	0.0	40.8	1,146.7	100.0	0.0	0.0	13.4
		0.5	725.5	100.0	0.0	0.0	37.7	725.5	100.0	0.0	0.0	30.7
		0.8	359.2	100.0	0.0	0.0	435.3	359.2	100.0	0.0	0.0	74.1
	0.6	0.2	2,853.9	100.0	0.0	0.0	6.3	2,853.9	100.0	0.0	0.0	4.8
		0.5	2,686.7	100.0	0.0	0.0	4.0	2,686.7	100.0	0.0	0.0	4.1
		0.8	2,429.4	100.0	0.0	0.0	9.6	2,429.4	100.0	0.0	0.0	6.8
Promedio			657.4	92.5	15.2	38.2	784.8	657.4	96.9	11.9	42.3	458.0

A partir de la información mostrada en la Tabla 5, se puede notar que hay instancias que no se pueden resolver dentro del límite de tiempo (valores de Gap superiores a 0) y, como era de esperarse, la cantidad de instancias que alcanzan el límite de tiempo aumentan, debido al aumento de la cantidad de tareas en la instancia. Para esos casos, los valores de Gap oscilan entre 2% y 100%. En otras palabras, incluso cuando el valor de la mejor solución encontrada VO es relativamente pequeño (por ejemplo, un valor de 1), si la cota inferior (valor de la relajación) CI es cero, el Gap reportado corresponderá al 100%. Es importante señalar que los valores de Gap tenderán a ser altos, ya que muchos de los valores para las relajaciones lineales (y por lo tanto el CI) para las instancias no resueltas son 0 o cercanas a 0, lo que sesga el valor promedio. El valor de la columna *RelGap* proporciona información adicional sobre cuánto se desvía el valor objetivo de la cota inferior (en términos de tardanza total)

Tabla 6: Resultados de la formulación para instancias de 100 trabajos

			MILP1					MILP2				
TiempoProc	α	β	TarTot	%Res	RelGap	%Gap	Tiempo	TarTot	%Res	RelGap	%Gap	Tiempo
[1,99]	0.2	0.2	97.8	75.0	3.0	3.1	3,222.2	97.8	95.0	5.0	2.8	1,327.1
		0.5	5.6	25.0	6.9	92.8	5,619.8	4.6	45.0	6.8	89.3	4,643.8
		0.8	5.0	65.0	10.2	78.5	2,847.9	4.6	70.0	9.7	69.9	2,461.8
	0.4	0.2	468.6	90.0	7.0	0.6	996.6	468.6	100.0	0.0	0.0	356.2
		0.5	304.7	30.0	34.5	10.8	6,293.6	302.9	45.0	33.1	9.7	5,981.3
		0.8	151.0	10.0	60.5	43.7	7,203.0	151.3	30.0	75.4	47.3	7,201.5
	0.6	0.2	1,098.8	95.0	9.0	0.3	527.0	1,098.8	100.0	0.0	0.0	301.7
		0.5	957.0	60.0	59.3	5.0	6,119.5	959.2	60.0	97.9	5.7	5,362.7
		0.8	899.9	45.0	192.5	24.6	6,757.6	902.6	45.0	236.7	23.6	6,359.0
[51,99]	0.2	0.2	96.9	100.0	0.0	0.0	180.6	97.5	100.0	0.0	0.0	62.2
		0.5	0.4	95.0	15.0	100.0	806.7	0.6	100.0	0.0	0.0	318.7
		0.8	1.4	100.0	0.0	0.0	89.1	1.7	100.0	0.0	0.0	29.2
	0.4	0.2	467.9	100.0	0.0	0.0	62.7	468.6	100.0	0.0	0.0	42.6
		0.5	277.6	90.0	23.0	0.7	1,503.4	280.6	100.0	0.0	0.0	155.9
		0.8	81.9	25.0	80.5	13.1	5,745.5	83.2	60.0	45.5	5.7	3,369.3
	0.6	0.2	1,098.4	100.0	0.0	0.0	81.7	1,098.8	100.0	0.0	0.0	30.0
		0.5	913.3	90.0	60.0	0.4	1,044.0	914.7	100.0	0.0	0.0	194.9
		0.8	697.3	75.0	449.6	3.5	1,946.1	695.2	75.0	412.4	3.2	1,897.3

Promedio			423.5	70.6	73.7	26.8	2,835.9	423.9	79.2	110.7	27.0	2,227.5

Tabla 7: Resultados de la formulación para instancias de 150 trabajos

			MILP1					MILP2				
TiempoProc	α	β	TarTot	%Res	RelGap	%Gap	Tiempo	TarTot	%Res	RelGap	%Gap	Tiempo
[1,99]	0.2	0.2	173.7	55.0	10.7	7.3	5,132.6	172.6	75.0	4.8	4.4	3,613.7
		0.5	6.1	30.0	8.7	100.0	5,445.6	4.6	45.0	8.0	96.8	4,644.0
		0.8	3.0	80.0	10.0	68.0	1,891.7	2.0	90.0	9.5	50.0	1,710.8
	0.4	0.2	881.9	70.0	9.8	0.7	3,589.0	881.7	85.0	6.7	0.3	2,183.9
		0.5	477.6	10.0	48.1	9.7	7,203.4	477.1	20.0	47.1	8.9	7,202.2
		0.8	202.4	10.0	121.3	60.8	7,202.2	191.5	0.0	107.2	56.0	7,201.5
	0.6	0.2	2,377.1	85.0	16.0	0.3	3,582.3	2,377.0	90.0	16.5	0.3	2,679.5
		0.5	1,933.2	60.0	217.5	10.7	7,203.3	1,933.6	40.0	157.5	9.5	7,202.8
		0.8	1,796.3	35.0	443.3	26.7	7,201.7	1,797.7	30.0	480.0	26.7	7,201.9
[51,99]	0.2	0.2	1,234.8	90.0	101.5	14.1	1,348.4	1,234.3	100.0	0.0	0.0	547.9
		0.5	0.0	100.0	0.0	0.0	1,086.1	0.0	100.0	0.0	0.0	134.3
		0.8	0.0	100.0	0.0	0.0	153.7	0.0	100.0	0.0	0.0	144.2
	0.4	0.2	8,242.0	90.0	97.5	2.3	1,113.8	8,241.7	100.0	0.0	0.0	219.0
		0.5	4,022.1	30.0	58.0	1.4	5,499.0	4,020.3	100.0	0.0	0.0	1,064.8
		0.8	733.5	0.0	210.1	28.6	7,202.6	703.0	10.0	124.0	16.5	6,680.0
	0.6	0.2	21,501.6	100.0	0.0	0.0	553.8	21,501.7	100.0	0.0	0.0	126.4
		0.5	17,050.4	70.0	68.2	0.3	2,910.5	17,050.2	90.0	30.0	0.1	1,380.3
		0.8	15,098.4	35.0	618.9	3.5	5,799.4	15,077.9	40.0	544.3	3.0	4,712.5
Promedio			4,207.4	58.3	137.1	22.0	4,117.7	4,203.7	67.5	133.0	21.9	3,258.3

De la información que se muestra en la Tabla 6 y la Tabla 7, se puede observar que MILP2 supera consistentemente a MILP1, reportando promedios más bajos de tardanza total (TarTot) y resolviendo de manera óptima un 9% más de instancias en promedio. Con respecto a los valores de %Gap, MILP1 reportó un promedio de RelGap levemente inferior, pero se puede explicar por el hecho de que los promedios fueron calculados considerando un número inferior de instancias no resueltas.

También es notable que MILP2 generalmente se desempeña mejor que MILP1, aunque ambos modelos reportan valores objetivos similares, el MILP2 requiere menos tiempo de cómputo. Específicamente, MILP2 reduce el tiempo medio de cómputo de 748.8 segundos a 458 segundos (reducción del 38.8%) en el caso de instancias de 50 trabajos, de 2,835.9 segundos a 2,227.5 segundos (reducción del 21.4%) en instancias de 100 trabajos, y de 4,117.7 segundos a 3,258.3 segundos (reducción del 20,8%) en instancias de 150 trabajo.

3.4.3. Resultados para el algoritmo.

El Algoritmo propuesto se codificó con Python 3.9.0. Los experimentos se realizaron en una PC con procesador Intel Xeon e5-1650 v2 @ 3.5 GHz y 8 GB de RAM, utilizando el sistema operativo Windows 10. Para dicho algoritmo, primero se realizó una experimentación preliminar para ajustar e identificar los mejores valores para los parámetros de operación del algoritmo. Luego de esto, se realizó la experimentación computacional que permitió evaluar el desempeño del algoritmo con respecto al MILP2 (para instancias pequeñas y medianas) y evaluar la consistencia en la calidad de las soluciones para instancias grandes. Se realizaron 10 ejecuciones del algoritmo para cada instancia con el fin de obtener el comportamiento promedio.

Los experimentos preliminares para la afinación de parámetros se realizaron utilizando un conjunto separado de instancias para cada combinación del grupo de tamaño mediano. El ILS se probó utilizando los cuatro tipos de estrategias (RND, EDD, TPI, MSL) en el procedimiento constructivo. Con respecto al parámetro $\boldsymbol{njr}$, se probaron tres valores diferentes $njr = \{0.2, 0.3, 0.4\}$. A partir de los resultados obtenidos, se estableció el valor de $njr = 0.3$, mientras que el valor de *Maxiter* se fijó en 10 dado que reportó el mejor equilibrio entre la calidad de las soluciones y el tiempo de CPU total transcurrido.

El primer análisis se realizó para evaluar la calidad de cada procedimiento constructivo, así como para determinar si existe una relación positiva entre la solución inicial y la mejor solución encontrada (es decir, si el procedimiento que reporta la mejor construcción inicial lleva a obtener la solución óptima). La Tabla 8 resume la tardanza total promedio obtenida por el procedimiento constructivo inicial, la tardanza total promedio obtenida después de ejecutar el procedimiento de mejora por primera vez y la tardanza total promedio de la mejor solución encontrada. Además, reporta el número promedio de iteraciones y el tiempo promedio total de CPU transcurrido que tomó el algoritmo para terminar su ejecución. Los resultados para las instancias fueron agrupados de acuerdo con tiempos de procesamiento y tamaño (como se indica en las columnas 1 y 2), mientras que las columnas 4 a 7 reportan los valores obtenidos por cada tipo de estrategia utilizada en el procedimiento constructivo (RND, EDD, MSL, TPI).

Tabla 8: Desempeño del algoritmo según el procedimiento de solución inicial

TiempoProc	Grupo		EDD	MSL	RND	TPI
[1,99]	Pequeñas	Solución del Proceso Constructivo	290.3	286.8	384.9	340.4
		Solución del Proceso de Mejora	193.8	193.5	193.6	195.2
		Mejor solución encontrada	180.4	180.1	179.8	180.1
		Número de Iteraciones	11.8	11.9	11.9	11.9
		Tiempo Transcurrido	0.2	0.2	0.3	0.2
	Medianas	Solución del Proceso Constructivo	1,362.5	1,436.5	2,410.0	2,324.0
		Solución del Proceso de Mejora	563.0	564.0	558.3	553.9
		Mejor solución encontrada	508.9	511.1	509.5	509.2
		Número de Iteraciones	15.9	16.0	16.1	15.4
		Tiempo Transcurrido	56.3	57.2	57.7	55.6
	Grandes	Solución del Proceso Constructivo	10,397.8	10,829.2	22,202.7	21,418.0
		Solución del Proceso de Mejora	3,384.7	3,373.3	3,341.7	3,339.9

		Mejor solución encontrada	3,168.2	3,167.8	3,166.5	3,172.1
		Número de Iteraciones	15.8	15.8	15.3	15.7
		Tiempo Transcurrido	1,544.9	1,555.0	1,530.5	1,546.6
[51,99]	Pequeñas	Solución del Proceso Constructivo	729.0	729.3	857.6	733.3
		Solución del Proceso de Mejora	566.0	564.7	568.6	567.5
		Mejor solución encontrada	536.7	537.2	536.9	537.8
		Número de Iteraciones	12.5	12.3	12.6	12.3
		Tiempo Transcurrido	0.2	0.2	0.3	0.2
	Medianas	Solución del Proceso Constructivo	7,239.6	7,182.2	9,257.1	7,356.7
		Solución del Proceso de Mejora	4,604.1	4,614.3	4,563.7	4,603.3
		Mejor solución encontrada	4,379.8	4,380.3	4,379.9	4,375.5
		Número de Iteraciones	12.2	12.0	12.1	12.0
		Tiempo Transcurrido	49.0	47.5	48.4	48.5
	Grandes	Solución del Proceso Constructivo	52,040.5	51,992.1	71,846.4	54,402.8
		Solución del Proceso de Mejora	31,126.1	31,114.0	30,771.2	31,048.7
		Mejor solución encontrada	30,087.5	30,104.3	30,054.5	30,085.2
		Número de Iteraciones	12.8	12.8	12.8	12.9
		Tiempo Transcurrido	1,510.6	1,546.0	1,553.1	1,553.5

A partir de los resultados de la Tabla 8, se puede observar que la estrategia aleatoria RND generalmente construye soluciones iniciales que reportan peores valores de Tardanza Total, pero el procedimiento de mejora es lo suficientemente robusto ya que produce soluciones de calidad para todas las diferentes estrategias de ordenamiento para la creación de la solución inicial. También se puede observar que la estrategia RND produce resultados ligeramente mejores que las demás una vez aplicado el proceso de mejora, lo que se puede explicar porque, a diferencia de EDD, TPI y MSS, la estrategia RND permite explorar un espacio de búsqueda más amplio que, a su vez, beneficia al procedimiento de búsqueda local en la primera iteración. También se puede notar que la primera ejecución del proceso de mejora produce una reducción significativa de la solución inicial, mientras que la mejora

obtenida del resto de iteraciones perturbaciones-mejora tiende a ser menos beneficiosa. También se nota como el algoritmo suele ejecutar en promedio entre 11 y 16 iteraciones de perturbación-mejora, indicando que, en promedio, la mejor solución se encuentra entre la 1ra y la 6ta iteración. Este análisis, junto con la comparación entre el ILS y MILP2 que se mostrará más adelante, confirma la robustez del ILS.

Finalmente, la Tabla 9 presenta una comparación detallada entre el ILS y el MILP2, mostrando el promedio de Tardanza Total obtenido por la formulación (columna TarTotMILP2) y el promedio de Tardanza Total obtenido por el algoritmo (columna TarTotILS). La desviación relativa (columna %Dev) en que el ILS difiere del MILP se calcula como $\%Desv = 100 * (ILSTarTot - MILP2TarTot)/MILP2TarTot$. También se muestra el tiempo computacional promedio correspondiente para MILP2 (columna Tiempo MILP2) y el algoritmo (Columna Tiempo ILS) en segundos. Cuando no hay información del MILP2, se muestra el símbolo "-".

Tabla 9: comparación entre ILS y MILP2

TProc	Grupo	n	TarTot MILP2	TarTot ILS	% Dev	Tiempo MILP2	Tiempo ILS
[1,99]	Pequeñas	14	138.1	139.2	0.8	0.4	0.1
		16	158.6	160.6	1.2	0.5	0.2
		18	188.9	190.7	1.0	1.4	0.3
		20	227.8	229.9	0.9	2.3	0.4
	Medianas	50	155.4	160.4	3.1	787.2	5.3
		100	443.4	461.4	3.9	3,777.2	38.1
		150	870.9	907.3	4.0	4,848.9	126.6
	Grandes	200	-	1,509.1	-	-	324.2
		300	-	2,928.4	-	-	1,206.8
		400	-	5,068.5	-	-	3,101.7
[51,99]	Pequeñas	14	400.3	404.9	1.1	0.2	0.1
		16	467.2	475.4	1.7	0.3	0.2
		18	587.8	597.6	1.6	0.3	0.3
		20	657.8	670.6	1.9	0.6	0.4

Medianas	50	1,159.4	1,247.7	7.1	128.8	3.9
	100	3,747.4	3,981.5	5.9	677.8	30.5
	150	7,536.5	7,907.4	4.7	1,667.7	110.6
Grandes	200	-	13,273.3	-	-	291.4
	300	-	28,202.7	-	-	1,166.8
	400	-	48,772.7	-	-	3,164.3

Como se mencionó anteriormente, el algoritmo muestra robustez, reportando valores promedio de %Dev que van desde 0.8% a 7.1%, lo que indica el muy buen desempeño del ILS en comparación con el MILP2. Con respecto al tiempo de ejecución, el ILS se ejecuta en una fracción del tiempo de CPU transcurrido del MILP2. Además, el algoritmo muestra consistencia en los tiempos de CPU transcurridos, ya que reporta tiempos de ejecución similares sobre ambas variaciones en los tiempos de procesamiento ([1,99] y [51,99]). Esto confirma que el ILS puede considerarse como una alternativa adecuada para resolver el problema de programación de máquinas paralelas no relacionadas con la minimización de tardanzas.

3.5. Conclusión del capítulo.

Se ha propuesto una formulación lineal entera mixta y un algoritmo de búsqueda local iterado para tratar con instancias de tamaño pequeño, mediano y grande. En cuanto a la formulación matemática, se incluyó un conjunto adicional de desigualdades para mejorar el rendimiento del modelo al descartar soluciones subóptimas factibles, lo que resultó exitoso.

De acuerdo con los experimentos realizados, se puede establecer una conclusión importante sobre el proceso de generación de instancias. Como se mostró, el cálculo de $\bar{p}$ mostrado anteriormente en la literatura puede generar instancias con mucha holgura en las fechas de vencimiento. Por lo tanto, se decidió

generar nuevas instancias con un parámetro más ajustado, $\bar{p} = C_{max}$ (el makespan óptimo), que se obtuvo usando otra formulación matemática (Apéndice B).

La formulación propuesta fue capaz de resolver instancias de hasta 150 puestos de trabajo. Para estos tamaños de instancias, el modelo requiere un promedio de tiempo transcurrido de hasta 68 minutos en instancias con un rango de tiempo de procesamiento entre U[1,99] y hasta 57 minutos para instancias con un rango de tiempo de procesamiento de U[51,99]. Las desigualdades válidas propuestas mejoraron el rendimiento del modelo al reducir su tiempo computacional y resolver más instancias de forma óptima.

Con respecto a la metaheurística propuesta, el procedimiento conserva tanto simplicidad como la generalidad. Su eficacia se comprobó mediante la comparación con los mejores resultados proporcionados por el modelo MILP2, logrando soluciones factibles de buena calidad para instancias de hasta 400 trabajos en tiempos computacionales razonables. Demostró ser robusto, obteniendo desviaciones medias dentro del 7,1% de las soluciones obtenidas por MILP2, y presentando un tiempo de ejecución competitivo. Finalmente, el algoritmo mostró ser menos sensible a las variaciones en los tiempos de procesamiento para todas las instancias.

4. Problema de máquinas paralelas no relacionadas con minimización de makespan, tiempos de preparación dependientes de la secuencia y recurso compartido.

El problema de máquinas paralelas con tiempos de preparación se presenta constantemente en las industrias, siendo dichos procesos de preparación un factor importante que merma la eficiencia y aumenta los costos de producción (*Allahverdi & Soroush 2008)*. Debido a esto, dichos sistemas han sido estudiado ampliamente en la literatura (*Allahverdi, Gupta & Aldowaisan 1999).* Existen variantes de estos sistemas en los que los procesos de preparación se ven limitados por algún recurso adicional para su correcta ejecución: en este capítulo se estudiara una variante especifica de estos sistemas, en los cuales se considera como recurso adicional un operador el cual pasara de maquina a máquina efectuando dichos procesos de preparación, generando para tal a su vez una secuencia de actividades. A dicho sistema se le denomina como $R \,/\, s_{ijk} \,/\, C_{max}$ con recurso compartido sin traslape, el cual se describe más a detalle en este capítulo.

En la literatura aún no se ha abordado específicamente el problema $R \,/\, s_{ijk} \,/\, C_{max}$ con recurso compartido sin traslape; estudios detallados pueden ser encontrados en los trabajos de *Allahverdi, et al. (2008)* y *Allahverdi (2015)* en cuanto a sistemas con procesos de preparación, mientras que, para sistemas con recursos adicionales, puede encontrarse un estudio detallado en el trabajo de *Edis, Oguz & Ozkarahan (2013)*. Variantes del sistema que más asemejan al problema estudiado en este capítulo pueden encontrarse en los trabajos de Vlk, Novak & Hanzalek (2019) y Vlk, et al. (2019), en los cuales se considera sistemas de máquinas paralelas idénticas, con asignación de tareas a máquinas fija y pre-definida, y con un recurso compartido sin traslape.

Como ya se mencionó en capítulos anteriores, se puede visualizar este tipo de sistemas en las industrias de prototipado mediante impresión 3D, donde consideraríamos a un empleado que monta herramentales y materiales de impresión en los equipos, dependiendo del tipo de tarea que cada maquina va a

realizar durante su secuencia, y dichos procesos de preparación podrían requerir menos tiempo en secuencias de tareas que utilizan los mismos herramentales, o más tiempo en secuencias donde las tareas requieran el cambio de herramental.

4.1. Definición formal del problema.

Las siguientes consideraciones describen el problema $R \,/\, s_{ijk} \,/\, C_{max}$ con recurso compartido sin traslape:

- Existen n tareas a resolver, las cuales pertenecen al conjunto N. Dichas tareas serán trabajadas por cualquiera de las m máquinas paralelas disponibles, las cuales pertenecen al conjunto M.

- Todas las máquinas están disponibles de manera continua, siendo cada una capaz de atender una sola tarea a la vez.

- Tanto las preparaciones como los procesamientos de las tareas no pueden ser interrumpidos una vez iniciados.

- Una vez terminada la preparación de una tarea, inmediatamente se deberá iniciar su procesamiento.

- Todas las tareas están disponibles en todo momento para ser procesadas y no hay restricciones de precedencia entre tareas.

- Cada tarea sucesora $k \in N$ tiene un tiempo de procesamiento p_{ik} asociado a la máquina $i \in M$.

- Existe un conjunto de tareas predecesoras $N_0 = N \cup \{0\}$, siendo 0 una tarea artificial necesaria para establecer el inicio de la secuencia.

- Existe un tiempo de preparación s_{ijk} que se realiza en la máquina i al terminar de procesar la tarea $j \in N_0$ para poder procesar la tarea k.

- Existe un recurso compartido (entendido como un operador), el cual realizara todos los procesos de preparación para todas las tareas en todas las máquinas
- El recurso compartido solo podrá realizar un proceso de preparación a la vez
- El objetivo consiste en minimizar el makespan C_{max}

4.2. Modelo lineal.

En los sistemas de secuenciación que son descritos mediante un MILP con el enfoque de secuencias directas, suelen usarse como referencias los momentos que sucede un evento importante para una tarea, especialmente para evitar traslapes entre tareas. Para este sistema en específico se tomarán en cuenta 3 diferentes situaciones (hitos del tiempo):

- A_k: El momento en que se inicia la preparación de la tarea k
- B_k: El momento en que se termina la preparación/inicia el procesamiento de la tarea k
- C_k: El momento en que se termina el procesamiento de la tarea k

Estos 3 hitos del tiempo servirán como variables de decisión de los modelos propuestos, resultando en una variante del modelo por cada tipo de variable de referencia, generando así los modelos 1A, 1B y 1C respectivamente, y de los cuales se determinaron versiones adicionales mediante la inclusión de variables que permitirán re expresar algunas restricciones, denominados 2A, 2B y 2C. Todos los modelos serán comparados en su desempeño en la sección de experimentación de este capítulo. A su vez, el recurso compartido será tratado como una máquina, y por lo tanto tendrá su propia secuencia de actividades, siendo éstas los procesos de preparación de las tareas. Para demostrar adecuadamente el uso de dichos hitos de tiempo se presenta a manera de ejemplo ilustrativo un sistema que considera 5

tareas y 2 máquinas, el cual contará con los siguientes tiempos de preparación y de procesamiento:

$$p_{ik} = \begin{bmatrix} 25 & 23 & 33 & 35 & 26 \\ 30 & 24 & 27 & 30 & 29 \end{bmatrix}$$

$$s_{1jk} = \begin{bmatrix} 3 & 4 & 2 & 3 & 2 \\ - & 7 & 5 & 10 & 7 \\ 6 & - & 9 & 13 & 9 \\ 8 & 6 & - & 4 & 9 \\ 11 & 9 & 9 & - & 7 \\ 7 & 8 & 12 & 6 & - \end{bmatrix}$$

$$s_{2jk} = \begin{bmatrix} 3 & 4 & 2 & 2 & 3 \\ - & 4 & 6 & 11 & 5 \\ 8 & - & 9 & 9 & 10 \\ 5 & 7 & - & 6 & 9 \\ 10 & 8 & 7 & - & 8 \\ 8 & 7 & 10 & 7 & - \end{bmatrix}$$

Los tiempos de procesamiento de la tarea k en la máquina i (p_{ik}) y los tiempos de preparación necesarios para cambiar cualquier tarea j por cualquier tarea k en la máquina 1(s_{1jk}) y en la máquina 2 (s_{2jk}). Se considerará para el ejemplo la siguiente asignación de tareas a las 2 máquinas disponibles y el recurso compartido:

$$SM_1 = \{5,2,1\}$$

$$SM_2 = \{3,4\}$$

$$SM_R = \{5,3,2,4,1\}$$

Donde SM_1 representa la secuencia de la máquina 1, SM_2 la secuencia de tareas a realizar en la máquina 2, y SM_R la secuencia en la que el recurso compartido ejecutará los procesos de preparación de todas las tareas en sus máquinas correspondientes. En este ejemplo se realizará en la máquina 1 primero la tarea 5, luego la 2 y por último la 1, mientras que en la máquina 2 se realizará primero la tarea 3 y luego la 4. En cuanto a la secuencia del recurso compartido, primero se realiza el proceso de preparación de la tarea 5 en la máquina 1, luego el de la tarea

3 en la máquina 2, posteriormente la tarea 2 en la máquina 1, luego la tarea 4 en la máquina 2, y por último la tarea 1 en la máquina 1.

Debido a la selección de secuencias en la solución propuesta, los tiempos de procesamiento y preparación para la solución del ejemplo ilustrativo serán: para la tarea 5 se tiene $s_{1,0,5} = 2$, $p_{1,5} = 26$, para la tarea 3 se tiene $s_{2,0,3} = 2$, $p_{2,3} = 27$, para la tarea 2 se tiene $s_{1,5,2} = 8$, $p_{1,2} = 23$, para la tarea 4 se tiene $s_{2,3,4} = 6$, $p_{2,4} = 30$, para la tarea 1 se tiene $s_{1,2,1} = 6$, $p_{1,1} = 25$. Esto puede visualizarse en el siguiente diagrama de Gantt (Figura 18):

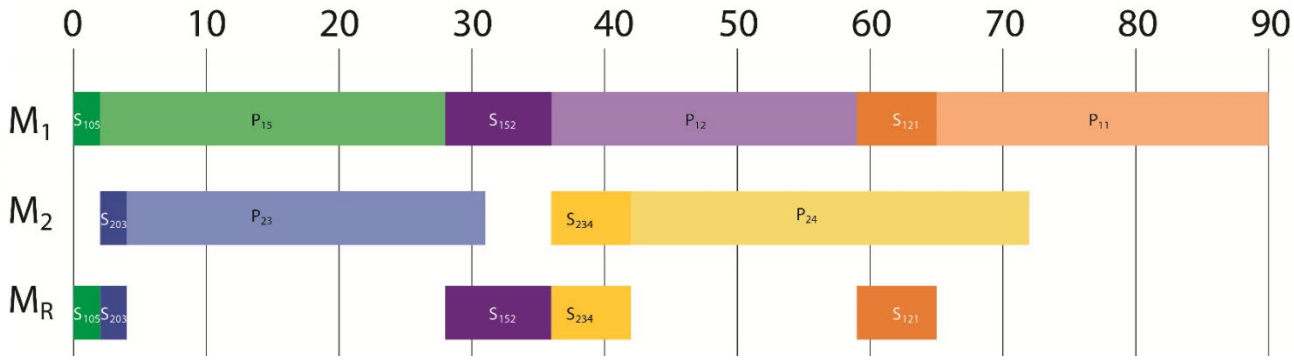


Figura 18: Diagrama de Gantt para la secuencia ilustrativa

Considerando lo mostrado en la Figura 18 se puede calcular para cada tarea el momento en que inicia su preparación A_k, el momento en que termina su preparación e inicia su procesamiento B_k, y el fin de su procesamiento B_k, siendo estos: para la tarea 5 $A_5 = 0, B_5 = 2, C_5 = 28$, para la tarea 2 $A_2 = 28, B_2 = 36, C_2 = 59$, para la tarea 1 $A_1 = 59, B_2 = 65, C_2 = 90$, para la tarea 3 $A_3 = 2, B_3 = 4, C_3 = 31$, y para la tarea 4 $A_4 = 36, B_4 = 42, C_4 = 72$, siendo el makespan $C_{max} = 90$.

Para los modelos presentados en este capítulo, en conjunto con las variables de referencia A_k, B_k, y C_k, anteriormente mencionadas, se considerarán también las variables de decisión binarias X_{ijk}, que tomarán el valor de 1 si la tarea k se asigna después de la tarea j en la máquina i, 0 en caso contrario. Para las asignaciones del recurso compartido se considerará Z_{jk}, que tomará el valor 1 si la preparación para iniciar la tarea k se asigna después de finalizar la preparación de la tarea j.

Dado que existirán diferentes versiones del modelo, primero se describirán todas las restricciones que son comunes para todas las versiones (así como la función objetivo), siendo las siguientes:

$$\text{Min } C_{max} \qquad \text{Eq.36}$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} = 1 \qquad \forall\, k \in N \qquad \text{Eq.37}$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} X_{ijk} \leq 1 \qquad \forall\, j \in N \qquad \text{Eq.38}$$

$$\sum_{\substack{h \in N_0 \\ j \neq h \neq k}} X_{ihj} \geq X_{ijk} \qquad \forall\, i \in M,\, j \in N, k \in N, j \neq k \qquad \text{Eq.39}$$

$$\sum_{k \in N} X_{i0k} \leq 1 \qquad \forall\, i \in M \qquad \text{Eq.40}$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} Z_{jk} = 1 \qquad \forall\, k \in N \qquad \text{Eq.41}$$

$$\sum_{\substack{k \in N \\ j \neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N \qquad \text{Eq.42}$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} X_{ijk} \leq C_{max} \qquad \text{Eq.43}$$

El grupo de ecuaciones Eq.37 asegura que, para cada tarea, solo exista un predecesor en la máquina en la que fue asignada. El grupo de ecuaciones Eq.38 asegurará que cada tarea tenga cuando mucho un sucesor en la máquina en la que fue asignada. El grupo de ecuaciones Eq.39 establece un flujo/conexión entre tareas creando así secuencias. El grupo de ecuaciones Eq.40 verifica que cada máquina tenga solo una tarea como inicio de su secuencia. El grupo de ecuaciones Eq.41

controla que, para cada tarea de la secuencia del recurso compartido, solo se tenga un predecesor, mientras que el grupo de ecuaciones Eq.42 limita a máximo un sucesor. La ecuación Eq.43 crea una cota inferior adecuada, acelerando el cálculo del algoritmo de ramificación y acotamiento. Esto es necesario debido a que la relajación lineal del modelo resultaría en valores muy bajos de makespan, y dado que el recurso compartido funge como cuello de botella, una aproximación del tiempo que se terminaría de procesar el último de sus procedimientos de preparación estaría muy cerca del makespan óptimo.

4.2.1. Modelos 1A, 1B y 1C.

El modelo 1A está compuesto por las Eq.36 a Eq.43, añadiendo las restricciones Eq.44, Eq.45, Eq.46, las cuales son exclusivas para dicho modelo, y que son:

$$A_k + \sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} (s_{ijk} + p_{ik}) X_{ijk} \leq C_{max} \qquad \forall\, k \in N \qquad \text{Eq.44}$$

$$A_k + \mu(1 - X_{ijk}) \geq A_j + p_{ij} + \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihj} X_{ihj} \qquad \forall\, i \in M, j \in N, k \in N, j \neq k \qquad \text{Eq.45}$$

$$A_k + \mu(1 - Z_{jk}) \geq A_j + \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihj} X_{ihj} \qquad \forall\, j \in N, k \in N, j \neq k \qquad \text{Eq.46}$$

La Eq.44 determina el makespan comparándolo con los tiempos de terminación de cada tarea. La Eq.45 evita los traslapes entre tareas asignadas a las máquinas mientras que la Eq.46 evita traslapes entre procedimientos de

preparación. El valor de la constante μ puede ser asignado como el valor del makespan de una solución factible cualquiera.

Respecto al modelo 1B, éste se compone de las Eq.36 a Eq.43, con la adición de las restricciones Eq.47, Eq.48, Eq.49, Eq.50, las cuales son exclusivas para dicho modelo, y que son:

$$B_k + \sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} p_{ik} X_{ijk} \leq C_{max} \qquad \forall\, k \in N \qquad \text{Eq.47}$$

$$B_k - s_{ijk} + \mu\left(1 - X_{ijk}\right) \geq B_j + p_{ij} \qquad \forall\, i \in M, j \in N, k \in N, j \neq k \qquad \text{Eq.48}$$

$$B_k - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\, i \in M,\ k \in N \qquad \text{Eq.49}$$

$$B_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihk} X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq B_j \qquad \forall\, j \in N, k \in N, j \neq k \qquad \text{Eq.50}$$

Las Eq.47, Eq.48, Eq.50 describen los mismos comportamientos que fueron modelados con las ecuaciones Eq.44, Eq.45, Eq.46 respectivamente, pero en el contexto de la variable B_k (el tiempo de terminación de la preparación), mientras que la Eq.49 es necesaria para la formulación 1B debido a que contempla como referencia el momento de inicio de procesamiento de las tareas (i.e., terminación de la preparación), el cual debe hacerse para la primer tarea en cada secuencia.

Finalmente, el modelo 1C se encuentra formulado por las Eq.36 a Eq.43, añadiendo las restricciones Eq.51, Eq.52, Eq.53, Eq.54, las cuales son exclusivas para dicho modelo, y que son:

$$C_k \leq C_{max} \qquad \forall\, k \in N \qquad \text{Eq.51}$$

$$C_k - p_{ik} - s_{ijk} + \mu(1 - X_{ijk}) \geq C_j \qquad \forall\, i \in M, j \in N, k \in N, j \neq k \qquad \text{Eq.52}$$

$$C_k - p_{ik} - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\, i \in M,\ k \in N \qquad \text{Eq.53}$$

$$C_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} (p_{ik} + s_{ihk}) X_{ihk} + \mu(1 - Z_{jk}) \geq C_j - \sum_{i \in M} p_{ij} X_{ijk} \qquad \forall\, j \in N, k \in N, j \neq k \qquad \text{Eq.54}$$

De manera similar al modelo B, las Eq.51, Eq.52, Eq.54 describen los mismos comportamientos que fueron modelados con las ecuaciones Eq.44, Eq.45, Eq.46 respectivamente, pero en el contexto de la variable C_k (tiempo de terminación del procesamiento), mientras que la Eq.53 es necesaria para la formulación 1C, debido a que contempla como referencia el momento de terminación del procesamiento de la tarea (momento en que fue completada), con lo cual es necesario asegurarse que se haga tanto la preparación como el procesamiento para la primer tarea en la secuencia.

4.2.2. Variables adicionales y modelos 2A, 2B y 2C.

Con base en en los trabajos de *Fanjul-Peyro, Ruiz & Perea (2019)* y *Avalos-Rosales & Cardona-Valdés(2022)* los cuales presentan relajaciones para modelos lineales de secuenciación, se implementaron ideas similares para los modelos 1A, 1B y 1C; obteniendo así modelos derivados usando variables de asignación adicionales, a los cuales nos referiremos como modelos 2A, 2B y 2C. Dichas variables de asignación serán nombradas como Y_{ik}, las cuales tomarán el valor de 1 cuando la tarea k sea asignada en la máquina i o 0 en caso contrario. A pesar de estar declaradas como variables binarias, éstas pueden ser introducidas al modelo expresadas como variables continuas, es decir $Y_{ik} \geq 0$, y conservarán su comportamiento binario por extensión de las variables X_{ijk}, mediante las

restricciones Eq.55, Eq.56 y Eq.57, que ligan el comportamiento de ambas, acotando los valores que pueden tomar las variables Y_{ik} a los valores que hayan tomado las variables X_{ijk}. Dichas restricciones son:

$$Y_{ik} = \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ k \in N \qquad \text{Eq.55}$$

$$Y_{ij} \geq \sum_{\substack{j \in N \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ j \in N \qquad \text{Eq.56}$$

$$\sum_{i \in M} Y_{ik} = 1 \qquad \forall\, k \in N \qquad \text{Eq.57}$$

Las Eq.55, Eq.56, Eq.57 re-expresan el comportamiento de las restricciones Eq.37, Eq.38, Eq.39 respectivamente mediante el uso de las variables Y_{ik}, resultando en una formulación más compacta, en términos de la cantidad de elementos diferentes de cero en la matriz de coeficientes de las restricciones. La Eq.55, Eq.56 son análogas a la restricción Eq.37, Eq.38, asegurando que cada tarea tenga solo un predecesor y máximo un sucesor, respectivamente, mientras que la restricción Eq.57 asegura que cada tarea este asignada a una sola máquina permitiendo, para este modelo, prescindir de la restricción Eq.39, que establece un flujo continuo de tareas en las secuencias. Adicionalmente, algunas restricciones pueden re-expresarse gracias a la inclusión de las variables Y_{ik}, siendo las Eq.47 y Eq. 53, re-expresadas por la Eq.58 y la Eq.59, como se muestra a continuación:

$$B_k + \sum_{i \in M} p_{ik} Y_{ik} \leq C_{max} \qquad \forall\, k \in N \qquad \text{Eq.58}$$

$$C_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} (p_{ik} + s_{ihk}) X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq C_j - \sum_{i \in M} p_{ij} Y_{ij} \qquad \forall\, j \in N, k \in N, j \neq k \qquad \text{Eq.59}$$

Para este caso, el modelo 2A estará compuesto por las restricciones Eq.36,Eq.40 a Eq.46, y Eq.55 a Eq.57. Por su parte, el modelo 2B estará compuesto por las restricciones Eq.36,Eq.40 a Eq.43, Eq.48 a Eq.50 y Eq.55 a Eq.58. Finalmente, el modelo 2C estará compuesto por las restricciones Eq.36,Eq.40 a Eq.43, Eq.51 a Eq.53, Eq.55 a Eq.57, y Eq.59. Para facilitar las lecturas de los modelos, pueden ser consultados en el apéndice C.

La Tabla 10 detalla la complejidad matemática para cada una de variantes de modelos propuestas. Cabe resaltar que las variantes del modelo 2A, 2B, y 2C, a pesar de tener más variables continuas, tendrán menos restricciones que las variantes 1A, 1B y 1C respectivamente.

Tabla 10: Tamaño de los modelos para cada variante propuesta

	1A	**1B**	**1C**	**2A**	**2B**	**2C**
Cantidad de Variables Binarias	$n^2m + n^2 + nm + n$	$n^2m + n^2 + nm + n$	$n^2m + n^2 + nm + n$	$n^2m + n^2 + nm + n$	$n^2m + n^2 + nm + n$	$n^2m + n^2 + nm + n$
Cantidad de Variables Continuas	n	n	n	$n + nm$	$n + nm$	$n + nm$
Cantidad de Restricciones	$2n^2m + n^2 + 2n + m + 2$	$2n^2m + n^2 + 2n + nm + m + 2$	$2n^2m + n^2 + 2n + nm + m + 2$	$n^2m + n^2 + 2n + 2nm + m + 2$	$n^2m + n^2 + 2n + 3nm + m + 2$	$n^2m + n^2 + 2n + 3nm + m + 2$

Finalmente, tomando en cuenta la información mostrada en la Tabla 10: Tamaño de los modelos para cada variante propuesta, se tiene la hipótesis de que el modelo podría presentar problemas para resolver instancias de gran tamaño. Es por ello que también se desea presentar una alternativa que permita lidiar con este

tipo de instancias en un tiempo computacional razonable. Dicha alternativa se presentará en la sección 4.3.

4.3. Algoritmo metaheurístico.

Como alternativa al uso del modelo de programación lineal, se propone el algoritmo metaheurístico de descenso por vecindarios variables (VND), el cual será descrito en esta sección. Como se demostrará en la sección 4.4. Experimentación, el algoritmo metaheurístico propuesto puede abordar instancias de mayor tamaño que el modelo de programación lineal propuesto en un tiempo computacional competitivo.

En esta sección se describirá la representación de la solución concebida para la ejecución del algoritmo, la estructura general del algoritmo, los distintos procedimientos que lo componen, y las distintas variantes utilizadas para su implementación.

4.3.1. Representación de la solución.

Para la ejecución del algoritmo se consideró el representar la respuesta en forma de un vector de secuencia asociado al recurso compartido, en el que cada posición del vector contendrá de 2 valores (tuplas), siendo dichos valores la información sobre cuál es la siguiente tarea para la que tiene que realizar el proceso de preparación que el recurso compartido debe atender, y en que máquina debe realizar se realizará dicha preparación. Este vector puede visualizarse de manera generalizada de la forma:

$$[[j_1, m_1], [j_2, m_2], [j_3, m_3], \dots [j_n, m_n]]$$

Donde j_l indica la tarea a realizar su preparación en el turno $l, l = 1 \dots n$ mientras que m_l indica la máquina en la que se va a realizar la preparación en el turno $l, l = 1 \dots n$.

En esta representación, se asignan turnos o posiciones a la ejecución de los procesos de preparación que deberá realizar el recurso compartido (operador), y a partir de estos se calculará los tiempos de completación de las mismas tareas; por lo tanto, el vector tendrá tantas posiciones o tuplas en la secuencia como tareas existan en el sistema. Dicha concepción resultará siempre en soluciones factibles puesto que se calcula a posteriori los tiempos de inicio y finalización más tempranos de los distintos procedimientos de las tareas. A continuación, se ilustra un vector de solución para el sistema de 5 tareas y 2 máquinas presentado en el ejemplo ilustrativo de la figura (Figura 18):

$$[[5,1],[3,2],[2,1],[4,2],[1,1]]$$

4.3.2. Estructura general del algoritmo propuesto.

Se propone un algoritmo de descenso por vecindario variable (RVND, Random Variable Neighborhood Descent) con un mecanismo de creación de una solución inicial codiciosa basada en lista de candidatos, y búsquedas locales mediante 4 mecanismos iterados con selección aleatoria. Dicho algoritmo tendrá 2 variantes como medios de diversificación, una consistente en procedimientos de destrucción-construcción multi-arranque y otra con procedimientos de perturbación de respuestas. La estructura general del algoritmo se presenta en la Figura 19:

```
   //Algoritmo de descenso de vecindario variable
   Datos: m, n, S, P
1  iter = 0
2  (A, C) ← Proceso Constructivo (m, n, S, P)
3  Mientras iter ≤ Maxiter hacer:
```

```
4     (A´, C´) ← Proceso de Búsqueda Local (A, C)
6     Si C´ < C* entonces:
7         (A*, C*) ← (A´, C´)
8         iter = 0
9     Fin
10    Si C´ ≥ C* entonces:
11        iter = iter + 1
12    Fin
13    (A, C) ← Proceso de Diversificacion (m, n, S, P, A*, C*)
14 Fin
   Resultado: A*, C*
```

Figura 19: Pseudocódigo General del Algoritmo

En la Figura 19 se muestra el pseudocódigo para estructura general del algoritmo, el cual requiere como datos de entrada: el número de trabajos n, el número de máquinas m, una matriz S de dimensión $n \times n$, en el cual cualquier elemento s_{jk} representa el proceso de preparación necesario para cambiar de la tarea j a la k, además de una matriz P de dimensión $m \times n$, en la que cualquier elemento p_{ik} denota el tiempo que requiere el trabajo k para ser procesado en la máquina i. Finalmente, se necesita una lista de tareas a secuenciar, denotada por el vector U.

Cualquier solución factible dada por el algoritmo se representará mediante una matriz A, que denotará el vector de tuplas usado para representar dicha solución, cuya dimensión es de $n \times 2$, y donde cada elemento a_{l1} representa la tarea a realizar el proceso de preparación por el recurso compartido en el turno l, mientras que el elemento a_{l2} representará en que máquina se realizará la preparación de la tarea denotada por a_{l1}. El parámetro C representa el makespan de la solución dada por A.

Como se muestra en la línea 2 de la Figura 19, el procedimiento constructivo recibe como entrada los parámetros $m, n, S\ y\ P$ para crear una solución inicial con la siguiente estructura: una matriz de asignación A, y un parámetro C que denota el

makespan de la solución. Una vez construida, la solución inicial pasa a la etapa de búsquedas locales iteradas. Posteriormente se verifica si la solución obtenida en la última iteración de búsqueda local supera la mejor solución encontrada hasta el momento, en caso afirmativo se actualiza la mejor solución y el contador de iteraciones se reinicia en 0, en caso contrario aumentará en 1 el contador de iteraciones. El ciclo de Diversificación - Búsqueda Local se aplicará hasta que se haya ejecutado un determinado número de iteraciones consecutivas (denotado por el parámetro $Maxiter$) sin encontrar una mejora.

4.3.3. Proceso constructivo.

El proceso constructivo genera una solución inician basado en la asignación de tareas seleccionadas de una lista de candidatos U. Dicha selección se realizará de manera aleatoria de entre los primeros ξ elementos de dicha lista, en la cual se ordenan las tareas de manera descendente según su tiempo de procesamiento promedio, tiempo de preparación promedio, o la suma de ambos, según sea indicado por el parámetro V, es decir, $V = \{p_{ik}, s_{ijk}, s_{ijk} + p_{ik}\}$, . El trabajo seleccionado se prueba en todas las posiciones existentes de A, para cada máquina en el sistema, verificando el correspondiente incremento en la tardanza del sistema $\Delta C = \widetilde{C_{max}} - C_{max}$, donde $\widetilde{C_{max}}$ representa el makespan de la solución anterior a ser modificada, para después asignar dicha tarea a la mejor posición y máquina probada. El procedimiento finaliza una vez se han asignado todas las tareas, es decir, cuando la lista de tareas pendientes por asignar (lista de candidatos U) quedó vacía. La Figura 20 muestra el pseudocódigo de este método constructivo.

```
//Proceso Constructivo
Datos: m, n, P, S, V, ξ
1 Inicializar A = ∅
2 Crear lista U según parámetro V
3 Ordenar U de manera ascendente según el parámetro V
4 Hacer:
```

```
5       Seleccionar al azar un trabajo j de entre las posiciones 1 y ξ de la lista U
6       Para cada máquina i hacer:
7           Para cada posición l en A hacer:
8               Calcular ΔC por asignar el trabajo j en la máquina i en la posición l
9           Fin
10      Fin
11      Asignar trabajo j en la máquina i en la posición l que produzca el menor ΔC
12      Eliminar trabajo j de U
13  Mientras U ≠ ∅
14  Calcular C
    Resultado: A, C
```

Figura 20: Pseudocódigo del Proceso Constructivo

El parámetro V intenta determinar tiempos relevantes del sistema para la creación de las listas de candidatos, y el ordenarlos de manera descendente resulta en la capacidad de probar más posiciones para las tareas que estén al final de la lista, las cuales contarán con tiempos más cortos. La selección de los 3 distintos valores que pude tomar el parámetro V implicará la ejecución completa de 3 distintas versiones para cada variante del algoritmo.

4.3.4. Mecanismos de búsqueda Local.

El mecanismo de búsqueda local incorpora métodos de descenso de vecindario variable (VND) al contar con 4 métodos distintos de creación vecindarios, los cuales exploran exhaustiva y codiciosamente, reiniciando las búsquedas aplicando el criterio de primera mejora. La estructura de dichos métodos es la siguiente:

Vecindario 1 - Reinserción: de manera similar al proceso constructivo, se crea una lista de candidatos U, la cual es ordenada de manera ascendente con base en los valores asociados al parámetro V, siendo en este caso los tiempos de la asignación actual A en lugar de los valores promedio. El proceso consiste en

seleccionar uno a uno los elementos (tareas) de la lista, para después extraerlos de la solución y probarlos en todas las otras posiciones de la solución. Esto con el propósito de intentar buscar una nueva posición y/o máquina en la cual dicha tarea tenga un mejor desempeño que pueda reducir el valor de la función objetivo(ΔC). En caso de encontrar una mejor posición de la tarea ($\Delta C < 0$), debido a que ha cambiado la solución, el proceso de reinserción deberá de reiniciarse recalculando la lista U; en caso de no encontrar una mejora con un elemento de la lista, se procederá a intentar mejorar el siguiente elemento de la lista. El proceso terminará una vez que se ha intentado mejorar todos los elementos de la lista U sin haber logrado una mejora. La Figura 21 muestra el pseudocódigo para este proceso.

```
   //Vecindario Proceso de Reinserción
   Datos: m, n, P, S, V, A
1  Crear lista U según parámetro V
2  Ordenar U de manera ascendente según el parámetro V
3  Hacer:
4      Seleccionar el siguiente trabajo j de la lista U
5      Guardar i, l de la tarea j en la solución actual A
6      Asignar ΔC = 0
7      Para cada máquina i hacer:
8          Para cada posición l en A hacer:
9              Calcular ΔC por asignar el trabajo j en la máquina i en la posición l
10             Guardar ΔC, i, l si son los mejores encontrados hasta el momento
11         Fin
12     Fin
13     Eliminar trabajo j de U
14     Si ΔC < 0 entonces:
15         Asignar trabajo j en la máquina i en la posición l en la solución actual A
16         Ir al renglón 1
17     Fin
18 Mientras U ≠ ∅
19 Calcular C
   Resultado: A, C
```

Figura 21: Pseudocódigo del Proceso de Reinserción

Vecindario 2 - Intercambio: en este procedimiento se intenta intercambiar tareas entre posiciones de tuplas, sin modificar la maquina en la tupla, por lo que este procedimiento hace las veces de intercambio de tareas entre secuencias de máquinas, e intercambio de tareas dentro de la secuencia de una misma máquina. Este procedimiento se realiza de manera exhaustiva, evaluando para cada tarea en cada posición el intercambio con todas las posiciones posteriores. Si en algún intercambio se encuentra una mejora ($\Delta C < 0$), se reiniciará el procedimiento. El procedimiento terminará cuando se hayan probado todos los intercambios posibles si encontrar alguna mejora. La Figura 22 muestra el pseudocódigo de este proceso.

```
   //Vecindario Proceso de Intercambio
   Datos: m, n, P, S, A
 1 Para cada j < n − 1:
 2     Para cada j' < n y j' > j :
 3         Probar intercambiar en A la tarea j y j'
 4         calcular ΔC
 5         Si ΔC < 0 entonces:
 6             Conservar el intercambio entre trabajo j y j' en la solución actual A
 7             Asignar j = 1, j' = 2 , Ir al renglón 1
 8         Fin
 9     Fin
10 Fin
11 Calcular C
   Resultado: A, C
```

Figura 22: Pseudocódigo del Proceso de Intercambio

Vecindario 3 - Reordenamiento: en este proceso se intercambian 2 tuplas de la solución, lo que resultará en un reordenamiento de la secuencia del recurso compartido y en el reordenamiento de las secuencias de las máquinas. Si en algún reordenamiento se encuentra una mejora ($\Delta C < 0$), se reiniciará el procedimiento. El procedimiento terminará cuando se hayan probado todos los reordenamientos

posibles sin encontrar alguna mejora. La Figura 23 muestra el pseudocódigo de este proceso.

```
   //Vecindario  Proceso de Reordenamiento
   Datos: m, n, P, S, A
 1 Para cada l < n − 1:
 2     Para cada l' < n y l' > l :
 3         Probar intercambiar en A la tupla de  l y l'
 4         calcular ΔC
 5         Si ΔC < 0 entonces:
 6             Conservar el intercambio entre la tupla l y l' en la solución actual A
 7             Asignar l = 1, l' = 2 , Ir al renglón 1
 8         Fin
 9     Fin
10 Fin
11 Calcular C
   Resultado: A, C
```

Figura 23: Pseudocódigo del Proceso de Reordenamiento

Vecindario 4 - Reasignación: en este proceso se probará cambiar la máquina de cada tupla, sin cambiar su posición en A, esto resultará en un cambio en el orden en el que el recurso compartido acudirá a las máquinas. De manera similar a los procesos anteriores, este proceso se efectuará de manera exhaustiva, reiniciando se encuentra una mejora y terminando solo cuando se efectúen todas las reasignaciones posibles en una solución sin encontrar alguna mejora. La Figura 24 muestra el pseudocódigo para este proceso.

```
   //Vecindario  Proceso de Reasignación
   Datos: m, n, P, S, A
 1 Para cada l ≤ n:
```

```
2     Para cada i ≤ m :
3         Probar en la tupla de la posición l  asignar la máquina i
4         calcular ΔC
5         Si ΔC < 0 entonces:
6             Conservar la asignación de la máquina i en la tupla l de la solución A
7             Asignar l = 1, i = 1 , Ir al renglón 1
8         Fin
9     Fin
10 Fin
11 Calcular C
   Resultado: A, C
```

Figura 24: Pseudocódigo del Proceso de Reasignación

Como se mencionó anteriormente, estos 4 procesos serán utilizados en el proceso de búsquedas locales por descenso de vecindarios variables (VND). Este mecanismo se aplica mediante la selección y aplicación de manera aleatoria de cualquiera de los procedimientos. Si dicho procedimiento no encontró una solución después de haber sido aplicado, será descartado y se procederá a seleccionar nuevamente un proceso de búsqueda local de entre los que aún no han sido descartados. En cambio, si el proceso resulta en una mejora, todos los procesos descartados serán nuevamente considerados para los procesos de selecciones aleatorias. Este proceso se terminará cuando todos los procesos de búsqueda local hayan sido descartados. La Figura 25 muestra el pseudocódigo de este proceso.

```
   //Proceso de descenso por vecindarios variables
   Datos: m, n, P, S, A
1  Inicializar Descartados = 0
2  Hacer:
3     Seleccionar al azar {Reinserción, Intercambio, Reordenamiento, Reasignación}
4     Si el proceso seleccionado resulto en una mejora entonces:
5        Recuperar los procesos descartados en iteraciones anteriores
```

```
6        Descartados = 0
7     Fin
8     Si el proceso seleccionado no resulto en una mejora entonces:
9        Descartar el proceso utilizado
10       Descartados+= 1
11    Fin
12 Mientras Descartados < 4
13 Calcular C
   Resultado: A, C
```

Figura 25: Pseudocódigo del Proceso de descenso de vecindarios variables

4.3.5. Mecanismo de diversificación.

Este procedimiento tiene como objetivo generar distintas soluciones iniciales para el procedimiento de búsqueda local (RVND), y para el cual se han creado 2 variantes según 2 distintas estrategias. Estas estrategias son:1) la estrategia multi arranque, en la cual se ejecuta múltiples veces el proceso constructivo para siempre tener una nueva solución al iniciar el proceso de búsqueda local, y 2) Perturbación aleatoria de la mejor solución encontrada al momento, con el propósito de proveer una nueva solución que conserva parte de la información de la mejor solución encontrada.

Para la estrategia multi arranque, la implementación en el algoritmo general solo implica ejecutar nuevamente el proceso constructivo, por lo que no se presentara un pseudocódigo para tal. Sin embargo, para la estrategia de perturbación se tendrán que especificar algunos parámetros. En este contexto, el parámetro ξ será utilizado para especificar la cantidad de tareas a reasignar. Se creará un alista de tareas aleatoriamente extraídas según el parámetro ξ, una vez extraídas de la solución, éstas serán insertadas nuevamente en la solución en la mejor posición y maquina posible, es decir, las tareas se incorporarán en la solución de tal manera que ΔC sea el menor posible. El pseudocódigo de este proceso se muestra en la Figura 26.

$//Proceso\ de\ perturbación$
Datos: A, ξ
1 Inicializar $lista = \emptyset$
Hacer:
Seleccionar arbitrariamente una tarea j de la solución A
Eliminar la tupla asociada a la tarea j de la solución A
Agregar j a $lista$
Mientras $|lista| < \xi$
Hacer:
Seleccionar el primer trabajo j de la $lista$
Para cada máquina i **hacer:**
Para cada posición l **en** A **hacer:**
Calcular ΔC por asignar el trabajo j en la máquina i en la posición l
Fin
Fin
Asignar trabajo j en la máquina i en la posición l que produzca el menor ΔC
Eliminar trabajo j de $lista$
Mientras $lista \neq \emptyset$
Resultado: A, C

Figura 26: Pseudocódigo para el procedimiento de perturbación

4.4. Experimentación.

En esta sección se describe todo lo referente a la experimentación llevada a cabo para evaluar el desempeño de las formulaciones y las versiones de los algoritmos propuestos. Específicamente, se describirán las metodologías para la creación de las instancias, los experimentos realizados, y los resultados obtenidos. Para evaluar el desempeño de los modelos y el algoritmos metaheurísticos, se crearon instancias usando tiempos de procesamiento p_{ik}, generados considerando una distribución uniforme discreta, $p_{ik} \sim U[1, 99]$ y tiempos de preparación s_{ijk}, generados considerando una distribución uniforme discreta, $s_{ijk} \sim U[1, 49]$, $s_{ijk} \sim U[1, 99]$, y $s_{ijk} \sim U[1, 124]$ según lo mostrado en *Avalos-Rosales, Alvarez & Angel-Bello* (*2013*), teniendo entonces el conjunto de distribuciones al que s_{ijk}

puede tomar valores dentro del siguiente intervalo $s_{ijk} \in \{[1,49],[1,99],[1,124]\}$. Con respecto a la cantidad de trabajos y máquinas, se consideraron varias combinaciones, esto con el fin de generar problemas de tamaño pequeño y mediano. El grupo de instancias pequeñas está formado por las posibles combinaciones de valores para $n \in \{10,12,14\}$ y $m \in \{2,3,4\}$, mientras que para las instancias de tamaño mediano se consideran combinaciones con los valores de $n \in \{20,40,60\}$ y $m \in \{2,4,6,8\}$. Se crearon 5 réplicas para cada instancia dentro de las combinaciones de p_{ik}, s_{ijk}, n, m, resultando en un total de 135 instancias pequeñas y 180 instancias medianas.

4.4.1. Resultados de las formulaciones matemáticas.

Respecto a las formulaciones lineales, éstas se codificaron utilizando el lenguaje AMPL y se resolvieron utilizando el motor GUROBI 9.1.0, estableciendo un tiempo máximo de ejecución de 3600 segundos (1 hora). Los experimentos se ejecutaron en una PC con procesador Intel i7 @ 2.4 GHz y 8 GB de RAM, utilizando el sistema operativo Windows 8.

Para evaluar el desempeño de las distintas formulaciones se consideró el grupo de instancias pequeñas, con el propósito de establecer cuál variante entrega resultados de mejor calidad. En la Tabla 11 se muestra una comparación general de las distintas variantes, mostrando el porcentaje de instancias en las que el modelo correspondiente logró alcanzar el valor óptimo global (denominado como %Opt), así como el tiempo promedio (en segundos), y el gap (en porcentaje) para las instancias no resueltas a optimalidad (denotado como %Gap).

Tabla 11: Comparación de resultados entre las diferentes variantes para el modelo lineal

Formulación	%Opt	Tiempo	%Gap
1A	79.3%	1066	31.1%
1B	82.2%	943	27.8%

1C	81.5%	1029	28.7%
2A	77.0%	1067	28.3%
2B	83.0%	955	28.0%
2C	81.5%	1039	29.3%
Promedio	80.7%	1017	28.9%

Como se puede observar en la Tabla 11, los modelos 1B y 2B presentan los tiempos más cortos de resolución promedio (943 y 955 segundos respectivamente). Además, cuentan con el mayor porcentaje de instancias resueltas a optimalidad (82.2% y 83%) y los valores de %Gap promedio más bajos (27.8% y 28%). Esto se puede atribuir a la forma en que se expresan las distintas ecuaciones del ambos modelos, que al utilizar la variable B_k, permiten calcular con mayor facilidad el momento en que se inició dicha preparación y el momento en que se termina el procesamiento de la tarea, relativo al tiempo de terminación del proceso de preparación (Eq.47 a Eq.50).

En la Tabla 12, se muestran estas mismas métricas, desglosados según los distintos tiempos de preparación.

Tabla 12: Comparación de resultados según tiempos de preparación

	$s_{ijk} = [1,49]$			$s_{ijk} =[1,99]$			$s_{ijk} =[1,124]$		
Formulación	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap
1A	64.4%	1542	35.0%	80.0%	986	25.9%	93.3%	670	25.8%
1B	71.1%	1363	34.2%	84.4%	868	22.6%	91.1%	599	16.2%
1C	66.7%	1470	31.4%	84.4%	931	24.5%	93.3%	687	25.5%
2A	62.2%	1606	34.1%	77.8%	980	21.3%	91.1%	615	21.2%
2B	66.7%	1518	30.8%	86.7%	822	23.1%	95.6%	527	21.4%
2C	64.4%	1628	31.8%	86.7%	923	26.8%	93.3%	565	20.8%
Promedio	65.9%	1521	32.9%	83.3%	918	24.0%	93.0%	610	21.8%

En la Tabla 12 se puede notar cómo de manera consistente para todos los modelos, el porcentaje de instancias resueltas aumenta a medida que se hace más amplio el rango de generación de los tiempos de preparación, mientras que el tiempo de ejecución de los modelos disminuye al igual que los valores del % de gap (calculado en la Eq.35). Esto es atribuible al hecho de que el aumento del rango de generación de los tiempos de procesamientos resulta en valores más grandes, obteniendo así una secuencia para el recurso compartido que tendrá menos tiempos ociosos, lo cual resultará en que la cota calculada por Eq.43 será más adecuada.

La Tabla 13 muestra, según la cantidad de tareas, el porcentaje de instancias en las que el modelo correspondiente logró alcanzar el valor óptimo global, así como el tiempo promedio, y el porcentaje de gap.

Tabla 13: Comparación de resultados según cantidad de tareas

	n =10			n =12			n =14		
Formulación	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap
1A	100.0%	40	0.0%	84.4%	939	31.6%	53.3%	2219	35.8%
1B	100.0%	35	0.0%	88.9%	771	33.6%	57.8%	2024	29.7%
1C	100.0%	40	0.0%	91.1%	844	35.1%	53.3%	2204	32.1%
2A	100.0%	45	0.0%	82.2%	930	28.1%	48.9%	2226	30.5%
2B	100.0%	34	0.0%	88.9%	761	25.2%	60.0%	2071	33.6%
2C	100.0%	48	0.0%	91.1%	894	34.4%	53.3%	2174	32.6%
Promedio	100.0%	40	0.0%	87.8%	857	31.3%	54.4%	2153	32.4%

En la Tabla 13 se hace evidente como el incremento de tareas aumenta considerablemente la complejidad del modelo, al aumentar los tiempos de ejecución de los modelos de manera considerable y reducir la cantidad de instancias resueltas

a optimalidad, esto debido al incremento que resulta en la cantidad de variables binarias del modelo, el cual aumenta de manera cuadrática de acuerdo con el aumento la cantidad de tareas.

La Tabla 14 muestra, según la cantidad de máquinas, el porcentaje de instancias en las que el modelo correspondiente logró alcanzar el valor óptimo global, así como el tiempo promedio, y el porcentaje de gap.

Tabla 14: Comparación de resultados según cantidad de máquinas

	$m=2$			$m=3$			$m=4$		
Formulación	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap	%Opt	Tiempo	%Gap
1A	55.6%	1951	33.8%	88.9%	854	27.1%	93.3%	392	19.8%
1B	57.8%	1810	30.0%	91.1%	719	23.7%	97.8%	301	2.9%
1C	62.2%	1871	34.8%	88.9%	759	22.5%	93.3%	458	4.7%
2A	51.1%	1934	31.2%	86.7%	838	22.0%	93.3%	428	19.8%
2B	64.4%	1733	33.3%	91.1%	661	21.4%	93.3%	472	8.4%
2C	62.2%	1811	34.0%	88.9%	797	23.4%	93.3%	508	12.6%
Promedio	58.9%	1852	32.9%	89.3%	771	23.3%	94.1%	426	11.4%

Se puede notar en la Tabla 14 cómo el aumento de las máquinas resulta en una menor dificultad para el modelo, haciéndose evidente en el aumento del porcentaje de instancias resueltas a optimalidad y en un decremento en los tiempos promedio de ejecución de los modelos. Esto es atribuible al hecho de que se están aumentándolos recursos del sistema y que, a pesar del incremento de variables y restricciones que esto causa, la adición de recursos facilitará la distribución y ejecución de tareas, obteniendo así regiones factibles más fáciles de explorar.

4.4.2. Resultados para el algoritmo.

El Algoritmo propuesto se codificó en Python 3.9.0. Los experimentos se realizaron en una PC con procesador Intel i7 @ 2.4 GHz y 8 GB de RAM, utilizando el sistema operativo Windows 8. La experimentación para el algoritmo se efectuó primero para las instancias pequeñas, con el propósito de poder establecer comparativas de desempeño con los modelos de programación lineal y establecer parámetros, y posteriormente para instancias grandes, con el propósito de establecer consistencia en el desempeño. Cada instancia se corrió 10 veces con el propósito de analizar el comportamiento promedio y la consistencia de los resultados.

Las instancias pequeñas se corrieron creando las listas de candidatos según los valores para $V = \{p_{ik}, s_{ijk}, s_{ijk} + p_{ik}\}$, $\xi = \{1,2,3,4,5,6,7,8,9,10\}$, $maxiter = 100$, y utilizando mecanismo de diversificación por perturbación (Pert) y multiarranque (Mult). El primer análisis se realizó para comparar el desempeño de los distintos métodos de creación de listas de candidatos según el parámetro V. La Tabla 15 muestra, para los distintos valores del parámetro V, los resultados obtenidos de C_{max} y tiempo (en segundos), agrupados por cantidad de tareas y distribuciones de tiempos de preparación, ordenados tales de una manera incremental (creciente) en complejidad.

Tabla 15: Comparación de desempeño para el parámetro V

		C_{max}			Tiempo		
n	s_{ijk}	$V = p_{ik}$	$V = s_{ijk}$	$V = s_{ijk} + p_{ik}$	$V = p_{ik}$	$V = s_{ijk}$	$V = s_{ijk} + p_{ik}$
10	[1,49]	183.30	182.94	183.17	1.59	1.65	1.59
	[1,99]	219.03	219.73	218.66	1.62	1.67	1.63
	[1,124]	240.74	241.54	242.15	1.77	1.72	1.74
12	[1,49]	196.38	196.87	197.16	2.99	3.06	3.00
	[1,99]	265.59	266.15	265.85	3.28	3.38	3.23
	[1,124]	272.17	273.36	274.47	3.24	3.40	3.27

14	[1,49]	226.06	226.03	226.14	5.16	5.30	5.17
	[1,99]	263.20	263.89	265.15	5.39	5.67	5.49
	[1,124]	309.15	311.13	311.09	5.87	5.97	5.87
Promedio		241.74	242.40	242.65	3.43	3.53	3.44

De los resultados de la Tabla 15 se puede observar que, tanto en tiempo del valor de la función objetivo, como en tiempos de ejecución del algoritmo, para las instancias de tamaño pequeño no representa una gran diferencia entre los métodos de creación de las listas de candidatos, tendiendo a ser ligeramente más eficiente el método de creación de listas por tiempo de procesamiento. En cuanto al desempeño del parámetro ξ , en la Tabla 16 se muestra un resumen de los valores obtenidos en la función objetivo (C_{max}) y se determina el beneficio porcentual que resulta al incrementar la cantidad de elementos en la lista de candidatos (cambio).

Tabla 16: Comparación de desempeño para ξ

	n=10		n=12		n=14	
ξ	C_{max}	Cambio	C_{max}	Cambio	C_{max}	Cambio
1	242.2	-	283.2	-	311.5	-
2	216.8	10.5%	248.3	12.3%	271.0	13.0%
3	213.0	1.7%	243.3	2.0%	265.2	2.2%
4	211.8	0.6%	241.3	0.8%	262.5	1.0%
5	210.9	0.4%	240.3	0.4%	261.3	0.5%
6	210.4	0.2%	239.9	0.2%	260.2	0.4%
7	210.2	0.1%	239.4	0.2%	259.6	0.2%
8	210.1	0.0%	239.2	0.1%	259.3	0.1%
9	210.1	0.0%	239.2	0.0%	259.0	0.1%
10	210.3	-0.1%	239.3	-0.1%	259.1	0.0%

De los resultados de la Tabla 16 podemos observar como el mayor beneficio se obtiene al aumentar de una lista con selección aleatoria entre 1 candidato ($\xi = 1$) a 2 candidatos ($\xi = 2$) , obteniendo un decremento del valor de la función objetivo que varía entre el 10.5% y 13%; cabe aclarar que el tener una lista de candidatos

según el parámetro $\xi = 1$ implica el utilizar siempre el primer elemento de la lista a una lista, dejando de lado la posibilidad de seleccionar aleatoriamente algún elemento, mientras que cambiar a $\xi \geq 2$ permitirá tener una lista de candidatos para realizar selecciones aleatorias, lo que permite diversificar y evitar estancamientos en vecindarios locales. También es destacable observar como el aumentar ξ a valores superiores a 3 resulta en mejoras despreciables en la función objetivo, e inclusive para valores como $\xi = 10$ puede resultar en peores soluciones. Con esto se puede concluir que la cantidad ideal para ξ oscilará entre el 20 y 30% de las tareas (dado que se observan las proporciones de 3/10 a 3/14).

En cuanto al desempeño del algoritmo en sus distintas configuraciones de diversificación se refiere, se muestra en la Tabla 17 una comparativa de su desempeño contra el de los mejores resultados obtenidos por las distintas variantes de los modelos de programación lineal propuestos (denominado conjuntamente como MILP). En dicha tabla se muestra el valor de la función objetivo obtenida por los modelos (C_{max} MILP), por el algoritmo de diversificación por multiarranque (C_{max} Mult), por el algoritmo de diversificación por perturbaciones (C_{max} Pert), así como la desviación porcentual de cada uno de los algoritmos respecto a la solución obtenida por los modelos lineales (MILP vs Mult y MILP vs Pert respectivamente), calculado estos últimos como las diferencias entre el valor de la función objetivo del algoritmo correspondiente con respecto al valor de la función objetivo del modelo lineal, dividido entre el valor de obtenido por el algoritmo.

Tabla 17: Comparación de desempeño entre MILP y Algoritmo

n	s_{ijk}	C_{max} MILP	C_{max} Mult	C_{max} Pert	MILP vs Mult	MILP vs Pert
10	[1,124]	234.33	240.17	242.78	2.43%	3.48%
	[1,99]	213.53	218.01	220.27	2.05%	3.06%
	[1,49]	179.67	182.78	183.50	1.70%	2.09%
12	[1,124]	262.33	272.62	274.04	3.77%	4.27%
	[1,99]	256.53	265.10	266.63	3.23%	3.79%
	[1,49]	190.67	196.19	197.41	2.82%	3.42%

14	[1,124]	291.60	309.63	311.28	5.82%	6.32%
	[1,99]	250.40	262.95	265.21	4.77%	5.58%
	[1,49]	217.47	226.32	225.84	3.91%	3.71%
Promedio		232.95	241.53	243.00	3.55%	4.13%

En la Tabla 17 se puede notar que los algoritmos, en ambas variantes muestran un desempeño adecuado y bastante similar para este tamaño de instancias, con valores de desviación que oscilan entre el 1.7% al 6.32% con respecto a la mejor solución encontrada por los modelos lineales. Cabe aclarar que, a pesar de parecer ligeramente superior el método multiarranque, la experimentación considerando instancias medianas demuestra que el algoritmo que incorpora perturbaciones en la diversificación suele tener mejor desempeño. La razón de que en la experimentación de instancias pequeñas el algoritmo multiarranque aparente ligeramente mejor comportamiento se debe a una susceptibilidad de la representación de la solución a la cantidad de tareas n. Las instancias pequeñas, al contar con menos tareas, tienden a resultar en perturbaciones de sus soluciones bastantes similares debido a la pequeña cantidad de posiciones en el vector de soluciones, por lo que crear una solución nueva mediante el mecanismo multiarranque resulta en soluciones más diversas para las instancias pequeñas.

En cuanto al desempeño en tiempos computacionales, la Figura 27 muestra como las distintas variantes de los modelos conservan tiempos razonables de procesamiento, que incrementan de manera moderada según incrementa la complejidad del problema. Por su parte, los modelos de programación lineal aumentan su tiempo de procesamiento de manera exponencial según el aumento de la complejidad del problema, principalmente por el aumento de la cantidad de tareas (como ya se mencionó antes). En la Figura 27 se muestra en el eje y, en escala logarítmica, el tiempo (en segundos), y en el eje x, las características de la instancia.

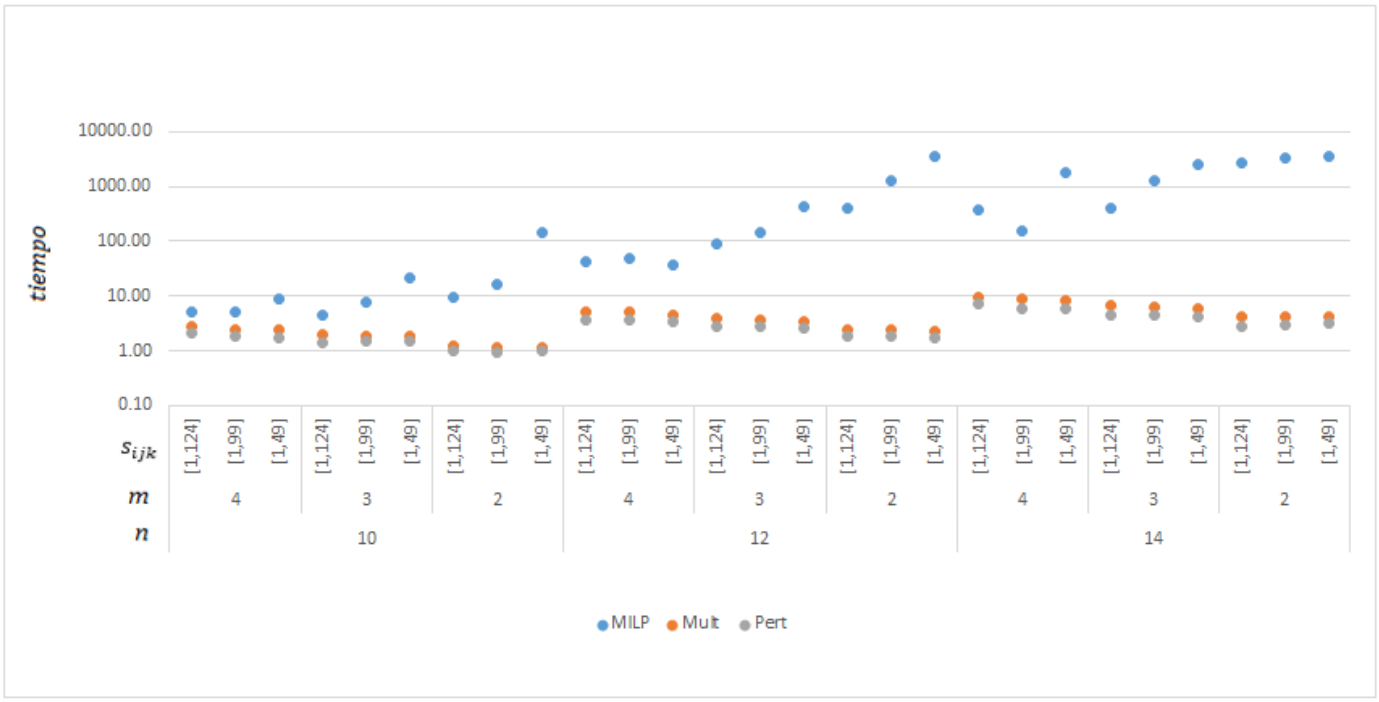


Figura 27: comparación de tiempos Algoritmos vs MILP

Las instancias medianas se corrieron creando las listas de candidatos según los valores para $V = \{p_{ik}, s_{ijk}, s_{ijk} + p_{ik}\}$, $\xi = \{\lfloor 0.2n \rfloor, \lfloor 0.3n \rfloor\}$, $maxiter = 100$, y utilizando mecanismo de diversificación por perturbación (Pert) y multiarranque (Mult). La razón de ejecutar para las instancias medianas con los parámetros de ξ, en este nuevo conjunto se debe a que durante la experimentación de instancias pequeñas se determinó que valores adecuados para la cantidad de tareas de entre las que se realiza la selección aleatoria de la lista de candidatos oscilan entre un 20 y 30 % de las tareas. Adicionalmente, se decidió redondear al entero menor el resultado de la multiplicación de n por los porcentajes, de ahí el uso de la función piso (denotada por los símbolos ⌊ ⌋). La Tabla 18 muestra los resultados obtenidos para las instancias medianas de manera análoga a lo mostrado en la Tabla 15. Es notable que, a pesar del incremento en los tamaños de las instancias, los tiempos de procesamiento siguen siendo moderados, e inclusive para las instancias de 60 tareas, dichos tiempos son competitivos en comparación a los obtenidos por los

modelos en instancias aproximadamente 4 veces más pequeñas. Y de manera similar a lo mostrado en las instancias medianas, los resultados obtenidos en la función objetivo al variar el parámetro V son similares, siendo $V = p_{ik}$ ligeramente sobresaliente.

Tabla 18: Comparación de desempeño para el parámetro V en instancias medianas

		C_{max}			Tiempo		
n	s_{ijk}	$V = p_{ik}$	$V = s_{ijk}$	$V = s_{ijk} + p_{ik}$	$V = p_{ik}$	$V = s_{ijk}$	$V = s_{ijk} + p_{ik}$
20	[1,49]	215.5	216.5	215.2	43.0	46.0	45.7
	[1,99]	285.1	283.2	281.7	41.6	47.9	49.0
	[1,124]	274.3	277.4	279.6	48.7	47.6	48.7
40	[1,49]	443.3	447.2	441.5	653.0	691.2	607.6
	[1,99]	505.6	507.2	500.4	723.3	728.6	729.7
	[1,124]	516.7	525.5	521.7	783.0	786.3	769.1
60	[1,49]	613.2	619.4	618.4	2919.8	3083.2	2678.6
	[1,99]	704.0	717.6	709.9	3609.5	3642.1	3317.4
	[1,124]	832.2	847.1	833.9	3316.2	3614.6	3718.9
promedio		487.7	493.4	489.1	1348.7	1409.7	1329.4

En cuanto a cómo se ve afectado el desempeño del algoritmo según la selección del parámetro ξ, se aplicaron valores más adecuados para la experimentación en instancias medianas, basado en los resultados obtenidos en las instancias pequeñas. Esto puedo visualizarse en la Tabla 19 donde se muestra para los distintos valores de ξ y de n, el valor promedio de la función objetivo y el tiempo promedio (en segundos).

Tabla 19: Comparación de desempeño para ξ en instancias medianas

	n=20		n=40		n=60	
ξ	C_{max}	Tiempo	C_{max}	Tiempo	C_{max}	Tiempo
$\lfloor 0.2n \rfloor$	260.9	42.0	489.7	665.4	716.8	3344.9
$\lfloor 0.3n \rfloor$	256.5	50.9	490.1	772.8	726.6	3299.6

En la Tabla 19 se puede observar que ninguno de los valores de ξ produce resultados significativamente superiores en el makespan promedio y tiempo promedio, y las pequeñas diferencias resultante no son consistentemente mejores para todos los tamaños de instancias, teniendo por ejemplo, que makespan es ligeramente más bajo para el valor de ξ=⌊0.3n⌋ en las instancias de tamaño n=20 y n=40, pero en las instancias de tamaño n=60 es más bajo el makespan de ξ=⌊0.2n⌋.

Por último, en la Figura 28 se muestra de manera gráfica el comportamiento de los algoritmos, mostrando el valor mínimo máximo y promedio de la función objetivo para ambas versiones de diversificación del algoritmo, según los diferentes tiempos de preparación y cantidad de tareas en la instancia.

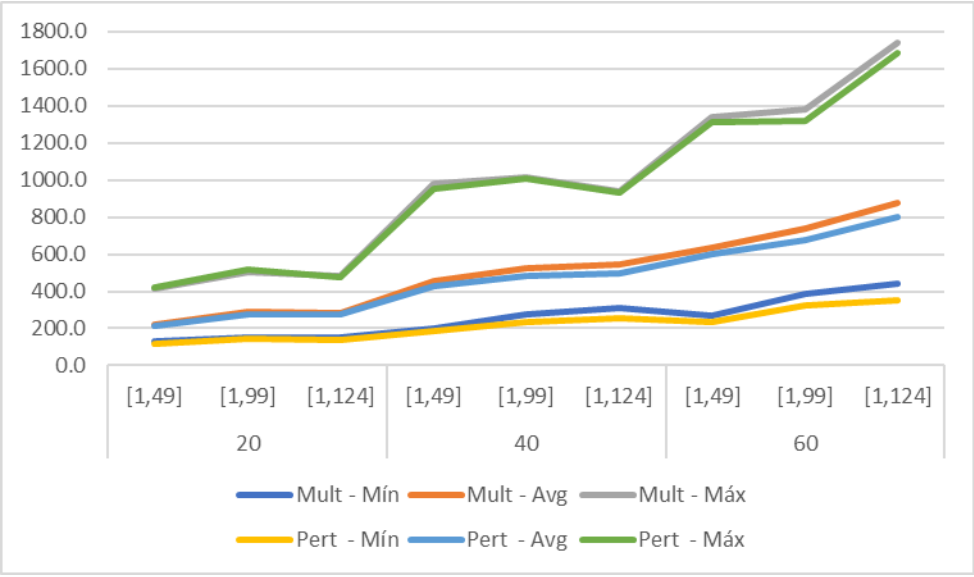


Figura 28: comparación consistencia de Algoritmos

En la Figura 28 se puede notar como las líneas de mínimo máximo y promedio correspondiente al algoritmo con perturbaciones están ligeramente más abajo que las del algoritmo multiarranque, esto sugiere que el algoritmo con perturbaciones tiene un desempeño ligeramente superior. La explicación para esto radica en que un proceso de perturbación robusto es capaz de dar diversidad al espacio de búsqueda del algoritmo, pero a su vez es capaz de conservar ciertas características de una buena solución.

Adicionalmente, en el Apéndice D se incluye a detalle los valores de máximo, promedio, mínimo, y desviación estándar, de manera desglosada para grupos de instancias, y se puede notar que la media es más cercana a los valores mínimos, así como valores de desviación estándar bajos, teniendo una media 507.8 en el makespan del algoritmo multiarranque, y una desviación estándar promedio de 12.5, resultando en un coeficiente de variación (CV) de 2.4%. Mientras que se tiene una media 475.5 en el makespan del algoritmo con perturbaciones, y una desviación estándar promedio de 20.2, resultando en un coeficiente de variación de 4.2%, lo que sugiere consistencia y robustez por parte de los algoritmos.

4.5. Conclusión del capítulo.

Se estudió el problema de secuenciación en máquinas paralelas no relacionadas con tiempos de preparación dependientes de la secuencia y recursos compartidos, considerando como función objetivo la minimización del makespan. Se presentaron 6 diferentes formulaciones lineales enteras mixtas y 2 versiones de un algoritmo de búsqueda de descenso de vecindario variable para tratar con instancias de tamaño pequeño, y mediano. En los modelos lineales se incluyeron funciones de cota inferior, así como variantes que consideran variables adicionales de asignación. Para el algoritmo se consideraron variantes de diversificación y se probaron distintos parámetros asociados a la operación del modelo.

De acuerdo con los experimentos realizados se ha concluido que las variantes 1B y 2B muestran mejor desempeño dentro de las 6 variantes evaluadas, y que la función de cota inferior facilita la exploración de la región factible, especialmente cuando los valores de los tiempos de preparación son mayores a los de procesamiento. Se observó que los modelos son susceptibles al aumento de la cantidad de tareas en la instancia, que hace aumentar la cantidad de variables enteras a razón cuadrática y por tanto no les es posible abordar instancias medianas.

En cuanto al algoritmo metaheurístico propuesto, muestra ser una alternativa viable al modelo en instancias pequeñas, donde obtiene valores de función objetivo cercano a los obtenidos a los modelos lineales, pero en tiempos computacionales mucho más cortos. Mientras que para instancias medianas muestra robustez, puesto que sus valores de media tienden a los valores mínimos, y sus valores de desviación estándar son pequeños (en relación a la media).

Dado que el problema estudiado en este capítulo aun es nuevo en la literatura, se carecen de puntos de comparación, por lo que puede ser factible abordar en trabajos futuros otras estrategias para resolver el problema, como lo podría ser desarrollar algoritmos poblacionales. Adicionalmente, se puede abordar el problema con múltiples recursos (operadores que realicen el proceso de preparación), con procesos de preparación que sean dependientes de la secuencia de actividades realizadas por el recurso compartido (y no necesariamente por la maquina donde fue procesada), con tiempos de liberación de tareas, o con tiempos de mantenimiento de máquinas.

5. Conclusiones de la tesis.

En esta tesis se estudiaron 2 problemas: El primero, fue el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza, y el segundo fue el problema de máquinas paralelas no relacionadas con minimización de makespan, tiempos de preparación dependientes de la secuencia y recurso compartido. Dichos problemas son relevantes para la industria, dado que ambos abordan problemas de actualidad en donde la administración adecuada de los recursos es crítica para mantener la competitividad.

Los objetivos generales del trabajo fueron cumplidos, ya que para ambos problemas presentados fue posible realizar modelos de programación lineal entera mixta que describieron de manera adecuada dichos problemas y que fueron capaces de obtener soluciones óptimas. Adicionalmente, para ambos problemas se presentaron algoritmos metaheurísticos que demostraron ser capaces de alcanzar soluciones de alta calidad en tiempos computacionales mejores que los de los modelos matemáticos, demostrando ser alternativas viables para la solución de los problemas presentados.

En cuanto a los objetivos específicos para el problema de secuenciación de máquinas paralelas no relacionadas con minimización de la tardanza, fueron alcanzados, como se explica a continuación:

- Se mostró un modelo de programación lineal entera mixta, capaz de encontrar óptimos globales en instancias de hasta 150 tareas. Además, se propuso la inclusión de desigualdades válidas, que permiten al modelo matemático reducir el tiempo de ejecución entre 20% y 38%.

- Se demostró que el modelo propuesto tiene un mejor desempeño que los modelos actuales de la literatura, probándose en instancias de 14 tareas, en las cuales el modelo propuesto fue capaz de resolver todas a optimalidad, en comparación contra un modelo previo de la literatura, el

cual presentó valores de gap promedio de hasta el 74%, de acuerdo con la experimentación realizada en este trabajo.

- Se mostró un algoritmo metaheurístico de búsquedas locales, que demostró ser significativamente más rápido que el modelo matemático propuesto, resolviendo las instancias de mayor tamaño abordadas por el modelo diez veces más rápido. El algoritmo demostró ser robusto con desviaciones de los valores óptimos no mayores al 7.1% y con la capacidad de resolver instancias de hasta 400 tareas.

- En cuanto a la metodología de creación de instancias prevalente en la literatura, se demostró es inadecuada para el problema en cuestión, puesto que se basa en estimaciones del makespan extrapoladas de sistemas de máquinas únicas, resultando en sistemas con holgura excesiva. Debido a esto, se propuso un cambio en tal metodología que permitió obtener instancias más razonables.

En cuanto a los objetivos específicos para problema de máquinas paralelas no relacionadas con minimización de makespan, tiempos de preparación dependientes de la secuencia y recurso compartido, estos también fueron alcanzados, y se pueden resumir de la siguiente manera:

- Se diseñaron 6 modelos de programación lineal entera mixta que adecuadamente describen el problema en cuestión, basándose en distintos hitos del tiempo relevantes para el sistema, resultando los modelos basados en el momento de terminación del proceso de preparación como hito de referencia (denominados como 1B y 2B) los que mostraron mejor desempeño. Todos los modelos fueron capaces de resolver instancias de hasta 14 tareas y hasta 4 máquinas.

- Los algoritmos propuestos fueron capaces de resolver de manera consistente todos los problemas abordados por el modelo lineal, logrando desviaciones con respecto al óptimo global no mayores al 6% y tiempos

de solución menores a 10 segundos, siendo hasta 40 veces más veloz que los modelos. Adicionalmente, los algoritmos fueron capaces de resolver instancias de hasta 60 tareas, mostrando consistencia en la calidad de la solución obtenida al presentar valores de desviación bajos.

En esta tesis se realizó un estudio detallado sobre los sistemas de secuenciación en máquinas paralelas, identificando adecuadamente el estado del arte y abordando 2 problemas específicos para los cuales se ha contribuido con nuevos modelos de programación lineal novedosos, así como algoritmos metaheurísticos competitivos. Para el primer problema, el modelo propuesto evita el uso de restricciones de gran M, lo cual no se encuentra en la literatura previa que aborda ese mismo problema, mientras que el algoritmo heurístico puede abordar instancias de gran tamaño, y demostró ser capaz de entregar soluciones competitivas en tiempos cortos en comparación con el modelo. El segundo problema es novedoso puesto que no se ha abordado anteriormente en la literatura y para el cual se diseñaron y compararon varios modelos matemáticos, enfatizando en todos ellos información relevante del sistema, así como algoritmos heurísticos que se presentan como alternativas viables a los modelos.

Como trabajos futuros se pretende abordar variantes del segundo problema, en el que se considerarán múltiples recursos adicionales, tiempos de preparación dependientes de la secuencia para las actividades del recurso compartido, tiempos de liberación de tareas y considerar tiempos de paro por mantenimiento de máquinas. También se pretende explorar otras metodologías de solución, como por ejemplo, algoritmos poblacionales (genéticos, evolutivos, búsqueda dispersa, colonia de hormigas, entre otros).

Referencias

Abdul-Razaq, T. S., Potts, C. N., & Van Wassenhove, L. N. (1990). A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete Applied Mathematics, 26(2-3), 235-253.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. European Journal of Operational Research, 246(2), 345-378.

Allahverdi, A., & Soroush, H. M. (2008). The significance of reducing setup times/setup costs. European Journal of Operational Research, 187(3), 978-984.

Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. Omega, 27(2), 219-239.

Allahverdi, A., Ng, C. T., Cheng, T. E., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. European journal of operational research, 187(3), 985-1032.

Anand, E., & Panneerselvam, R. (2015). Literature review of open shop scheduling problems. Intelligent Information Management, 7(01), 33.

Avalos-Rosales, O., & Cardona-Valdés, Y. (2022) Binary Variable Relaxation in MIP Formulations. Available at SSRN 4160695.

Avalos-Rosales, O., Alvarez, A., & Angel-Bello, F. (2013, June). A reformulation for the problem of scheduling unrelated parallel machines with sequence and machine dependent setup times. In Proceedings of the International Conference on Automated Planning and Scheduling (Vol. 23, No. 1).

Avalos-Rosales, O., Angel-Bello, F., & Alvarez, A. (2015). Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. The International Journal of Advanced Manufacturing Technology, 76(9-12), 1705-1718.

Azizoglu, M., & Kirca, O. (1998). Tardiness minimization on parallel machines. International Journal of Production Economics, 55(2), 163-168.

Balas, E. (1985). On the facial structure of scheduling polyhedra. In Mathematical Programming Essays in Honor of George B. Dantzig Part I (pp. 179-218). Springer, Berlin, Heidelberg.

Blazewicz, J., Dror, M., & Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. European Journal of Operational Research, 51(3), 283-300.

Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2019). Handbook on scheduling. Cham: Springer International Publishing.

Boussaïd, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. Information sciences, 237, 82-117.

Bruno, J., Coffman Jr, E. G., & Sethi, R. (1974). Scheduling independent tasks to reduce mean finishing time. Communications of the ACM, 17(7), 382-387.

Centeno, G., & Armacost, R. L. (1997). Parallel machine scheduling with release time and machine eligibility restrictions. Computers & industrial engineering, 33(1-2), 273-276.

Chaudhry, I. A., & Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research, 23(3), 551-591.

Cheng, C. Y., & Huang, L. W. (2017). Minimizing total earliness and tardiness through unrelated parallel machine scheduling using distributed release time control. Journal of manufacturing systems, 42, 1-10.

Cheng, T. C. E., & Sin, C. C. S. (1990). A state-of-the-art review of parallel-machine scheduling research. European Journal of Operational Research, 47(3), 271-292.

Cho, H., & Easwaran, A. (2020). Flow network models for online scheduling real-time tasks on multiprocessors. IEEE Access, 8, 172136-172151.

Conforti, M., Cornuéjols, G., & Zambelli, G. (2014). Integer programming (Vol. 271). Berlin: Springer.

De CM Nogueira, J. P., Arroyo, J. E. C., Villadiego, H. M. M., & Gonçalves, L. B. (2014). Hybrid GRASP heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties. Electronic Notes in Theoretical Computer Science, 302, 53-72.

Diana, R. O. M., de França Filho, M. F., de Souza, S. R., & de Almeida Vitor, J. F. (2015). An immune-inspired algorithm for an unrelated parallel machines' scheduling problem with sequence and machine dependent setup-times for makespan minimisation. Neurocomputing, 163, 94-105.

Du, J., & Leung, J. Y. T. (1990). Minimizing total tardiness on one machine is NP-hard. Mathematics of operations research, 15(3), 483-495.

Dyer, M. E., & Wolsey, L. A. (1990). Formulating the single machine sequencing problem with release dates as a mixed integer program. Discrete Applied Mathematics, 26(2-3), 255-270.

Vallada, E., Ruiz, R., & Minella, G. (2008). Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics. Computers & Operations Research, 35(4), 1350-1373.

Edis, E. B., & Ozkarahan, I. (2012). Solution approaches for a real-life resource-constrained parallel machine scheduling problem. The International Journal of Advanced Manufacturing Technology, 58(9), 1141-1153.

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. European Journal of Operational Research, 230(3), 449-463.

Emmons, H. (1969). One-machine sequencing to minimize certain functions of job tardiness. Operations Research, 17(4), 701-715.

Fanjul-Peyro, L. (2020). Models and an exact method for the Unrelated Parallel Machine scheduling problem with setups and resources. Expert Systems with Applications: X, 5, 100022.

Fanjul-Peyro, L., Perea, F., & Ruiz, R. (2017). Models and matheuristics for the unrelated parallel machine scheduling problem with additional resources. European Journal of Operational Research, 260(2), 482-493.

Fanjul-Peyro, L., Ruiz, R., & Perea, F. (2019). Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times. Computers & Operations Research, 101, 173-182.

Gedik, R., Kalathia, D., Egilmez, G., & Kirac, E. (2018). A constraint programming approach for solving unrelated parallel machine scheduling problem. Computers & Industrial Engineering, 121, 139-149.

Glover, F. (1997). Tabu search and adaptive memory programming—advances, applications and challenges. Interfaces in computer science and operations research, 1-75.

Gordon, V., Proth, J. M., & Chu, C. (2002). A survey of the state-of-the-art of common due date assignment and scheduling research. European Journal of Operational Research, 139(1), 1-25.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Kan, A. R. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. In Annals of discrete mathematics (Vol. 5, pp. 287-326). Elsevier.

Graves, S. C. (1981). A review of production scheduling. Operations research, 29(4), 646-675.

Gupta, S. K., & Kyparisis, J. (1987). Single machine scheduling research. Omega, 15(3), 207-227.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. European journal of operational research, 130(3), 449-467.

Hariri, A. M. A., & Potts, C. N. (1991). Heuristics for scheduling unrelated parallel machines. Computers & operations research, 18(3), 323-331.

Ho, J. C., & Chang, Y. L. (1991). Heuristics for minimizing mean tardiness for m parallel machines. Naval Research Logistics (NRL), 38(3), 367-381.

Jackson, J. R. (1955). Scheduling a production line to minimize maximum tardiness. CALIFORNIA UNIV LOS ANGELES NUMERICAL ANALYSIS RESEARCH.

Kan, A. R. (2012). Machine scheduling problems: classification, complexity and computations. Springer Science & Business Media.

Kedad-Sidhoum, S., Solis, Y. R., & Sourd, F. (2008). Lower bounds for the earliness–tardiness scheduling problem on parallel machines with distinct due dates. European Journal of Operational Research, 189(3), 1305-1316.

Keha, A. B., Khowala, K., & Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. Computers & Industrial Engineering, 56(1), 357-367.

Komaki, G. M., Sheikh, S., & Malakooti, B. (2019). Flow shop scheduling problems with assembly operations: a review and new trends. International Journal of Production Research, 57(10), 2926-2955.

Kyparisis, G. J., & Koulamas, C. (2006). Flexible flow shop scheduling with uniform parallel machines. European journal of operational research, 168(3), 985-997.

Lambora, A., Gupta, K., & Chopra, K. (2019, February). Genetic algorithm-A literature review. In 2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon) (pp. 380-384). IEEE.

Lasserre, J. B., & Queyranne, M. (1992). Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling.

Lawler, E. L., & Labetoulle, J. (1978). On preemptive scheduling of unrelated parallel processors by linear programming. Journal of the ACM (JACM), 25(4), 612-619.

Lee, J. Y., & Kim, Y. D. (2012). Minimizing the number of tardy jobs in a single-machine scheduling problem with periodic maintenance. Computers & Operations Research, 39(9), 2196-2205.

Lee, Y. H., & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence-dependent setup times. European Journal of Operational Research, 100(3), 464-474.

Lenstra, J. K., Kan, A. R., & Brucker, P. (1977). Complexity of machine scheduling problems. In Annals of discrete mathematics (Vol. 1, pp. 343-362). Elsevier.

Leung, J. Y. (Ed.). (2004). Handbook of scheduling: algorithms, models, and performance analysis. CRC press.

Li, K., & Yang, S. L. (2009). Non-identical parallel-machine scheduling research with minimizing total weighted completion times: Models, relaxations and algorithms. Applied mathematical modelling, 33(4), 2145-2158.

Liaw, C. F., Lin, Y. K., Cheng, C. Y., & Chen, M. (2003). Scheduling unrelated parallel machines to minimize total weighted tardiness. Computers & Operations Research, 30(12), 1777-1789.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: a survey. Computers & industrial engineering, 37(1-2), 57-61.

Logendran, R., McDonell, B., & Smucker, B. (2007). Scheduling unrelated parallel machines with sequence-dependent setups. Computers & Operations Research, 34(11), 3420-3438.

Lourenço, H. R., Martin, O. C., & Stützle, T. (2019). Iterated local search: Framework and applications. In Handbook of metaheuristics (pp. 129-168). Springer, Cham.

Malik, H., Iqbal, A., Joshi, P., Agrawal, S., & Bakhsh, F. I. (Eds.). (2021). Metaheuristic and evolutionary computation: algorithms and applications (pp. 46-61). Springer Nature Singapore.

Min, L., & Cheng, W. (1999). A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artificial Intelligence in Engineering, 13(4), 399-403.

Mokotoff, E. (2001). Parallel machine scheduling problems: A survey. Asia-Pacific Journal of Operational Research, 18(2), 193.

Nagar, A., Haddock, J., & Heragu, S. (1995). Multiple and bicriteria scheduling: A literature survey. European journal of operational research, 81(1), 88-104.

Panwalkar, S. S., & Iskander, W. (1977). A survey of scheduling rules. Operations research, 25(1), 45-61.

Parente, M., Figueira, G., Amorim, P., & Marques, A. (2020). Production scheduling in the context of Industry 4.0: review and trends. International Journal of Production Research, 58(17), 5401-5431.

Potts, C. N., & Van Wassenhove, L. N. (1982). A decomposition algorithm for the single machine total tardiness problem. Operations Research Letters, 1(5), 177-181.

Potvin, J. Y., & Gendreau, M. (Eds.). (2018). Handbook of Metaheuristics. Berlin/Heidelberg, Germany: Springer.

Queyranne, M., & Schulz, A. S. (1994). Polyhedral approaches to machine scheduling. Berlin: TU, Fachbereich 3.

Queyranne, M., & Wang, Y. (1991). Single-machine scheduling polyhedra with precedence constraints. Mathematics of Operations Research, 16(1), 1-20.

Radhakrishnan, S., & Ventura, J. A. (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. International Journal of Production Research, 38(10), 2233-2252.

Rafael M, Pardalos PM, Resende MG (eds) (2018) Handbook of Heuristics. Springer

Rajakumar, R., Dhavachelvan, P., & Vengattaraman, T. (2016, October). A survey on nature inspired meta-heuristic algorithms with its domain specifications. In 2016 international conference on communication and electronics systems (ICCES) (pp. 1-6). IEEE.

Resende, M. G., & Ribeiro, C. C. (2010). Greedy randomized adaptive search procedures: Advances and applications. Handbook of metaheuristics, 146, 281-317.

Reza Hejazi, S., & Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: a review. International Journal of Production Research, 43(14), 2895-2929.

Ruiz, R., & Andrés-Romano, C. (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. The International Journal of Advanced Manufacturing Technology, 57(5-8), 777-794.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. European journal of operational research, 205(1), 1-18.

Sadfi, C., Penz, B., Rapine, C., Błażewicz, J., & Formanowicz, P. (2005). An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints. European journal of operational research, 161(1), 3-10.

Schulz, A. S. (1996, June). Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In International conference on integer programming and combinatorial optimization (pp. 301-315). Springer, Berlin, Heidelberg.

Sen, T., Sulek, J. M., & Dileepan, P. (2003). Static scheduling research to minimize weighted and unweighted tardiness: a state-of-the-art survey. International Journal of Production Economics, 83(1), 1-12.

Sgall, J. (1998). On-line scheduling. Online algorithms, 196-231.

Shim, S. O., & Kim, Y. D. (2007). Minimizing total tardiness in an unrelated parallel-machine scheduling problem. Journal of the Operational Research Society, 58(3), 346-354.

Shim, S. O., & Kim, Y. D. (2007)b. Scheduling on parallel identical machines to minimize total tardiness. European Journal of Operational Research, 177(1), 135-146.

Sousa, J. P., & Wolsey, L. A. (1992). A time indexed formulation of non-preemptive single machine scheduling problems. Mathematical programming, 54(1), 353-367.

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. Journal of the operational research society, 57(10), 1143-1160.

Tanaka, S., & Araki, M. (2008). A branch-and-bound algorithm with Lagrangian relaxation to minimize total tardiness on identical parallel machines. International Journal of Production Economics, 113(1), 446-458.

Tran, T. T., Araujo, A., & Beck, J. C. (2016). Decomposition methods for the parallel machine scheduling problem with setups. INFORMS Journal on Computing, 28(1), 83-95.

Unlu, Y., & Mason, S. J. (2010). Evaluation of mixed integer programming formulations for non-preemptive parallel machine scheduling problems. Computers & Industrial Engineering, 58(4), 785-800.

Vallada, E., & Ruiz, R. (2011). A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. European Journal of Operational Research, 211(3), 612-622.

Vallada, E., & Ruiz, R. (2012). Scheduling unrelated parallel machines with sequence dependent setup times and weighted earliness–tardiness minimization. In Just-in-Time Systems (pp. 67-90). Springer, New York, NY.

Ventura, J. A., & Kim, D. (2000). Parallel machine scheduling about an unrestricted due date and additional resource constraints. Iie Transactions, 32(2), 147-153.

Villa, F., Vallada, E., & Fanjul-Peyro, L. (2018). Heuristic algorithms for the unrelated parallel machine scheduling problem with one scarce additional resource. Expert Systems with Applications, 93, 28-38.

Vlk, M., Novak, A., & Hanzalek, Z. (2019). Makespan Minimization with Sequence-dependent Non-overlapping Setups. In ICORES (pp. 91-101).

Vlk, M., Novak, A., Hanzalek, Z., & Malapert, A. (2019, February). Non-overlapping Sequence-Dependent Setup Scheduling with Dedicated Tasks. In

International Conference on Operations Research and Enterprise Systems (pp. 23-46). Springer, Cham.

Wenzelburger, P., & Allgöwer, F. (2019). A novel optimal online scheduling scheme for flexible manufacturing systems. IFAC-PapersOnLine, 52(10), 1-6.

Wolsey, L. A., & Nemhauser, G. L. (1999). Integer and combinatorial optimization (Vol. 55). John Wiley & Sons.

Yang, S. J., & Yang, D. L. (2010). Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities. Omega, 38(6), 528-533.

Apéndice A

La formulación de secuencias directas que se presenta a continuación fue adaptada del trabajo presentado por *De CM Nogueira, et al. (2014)*. Las variables de decisión son las siguientes:

$C_{ik} = \ representara\ el\ tiempo\ de\ completacion\ de\ la\ tarea\ i\ en\ la\ maquina\ k$

$T_i = tardanza\ del\ trabajo\ i$

$Y_{ijk} = 1\ si\ la\ tarea\ i\ precede\ a\ la\ j\ en\ la\ máquina\ k, 0\ en\ caso\ contrario$

La formulación es la siguiente:

$$\min \sum_{i \in K} T_i$$

$$\sum_{i \in M} \sum_{i \in N_0, i \neq j} Y_{ijk} = 1 \qquad \forall\, j \in N$$

$$\sum_{j \in N} Y_{0jk} \leq 1 \qquad \forall k \in M$$

$$\sum_{i \in N_0, i \neq h} Y_{ihk} \geq \sum_{j \in N, i \neq j} Y_{hjk} \qquad \forall\, h \in N, k \in M$$

$$C_{0k} = 0 \qquad \forall\, k \in M$$

$$C_{jk} \geq C_{ik} - \mu + \left(P_{jk} + \mu\right) Y_{ijk} \qquad \forall\, i \in N_0, j \in N, k \in M, i \neq j$$

$$T_i \geq C_{ik} - d_i \qquad \forall\, i \in N, k \in M$$

$$T_i \geq 0 \qquad \forall\, i \in N$$

$$C_{ik} \geq 0 \qquad \forall\, i \in N, k \in M$$

$$Y_{ijk} = \{0,1\} \qquad \forall\, i \in N_0, j \in N, k \in M$$

Apéndice B

En esta sección, se presenta la formulación implementada para calcular la duración óptima utilizada para estimar las fechas de vencimiento de los trabajos para las instancias usadas en la experimentación de la sección 3 (Figura 16). Las variables de decisión son:

$$Yik = 1\ si\ el\ trabajo\ i\ se\ asigna\ a\ la\ máquina\ k, 0\ en\ caso\ contrario.$$

Formulación Matemática:

$$\min C_{max}$$

$$\sum_{k \in M} Y_{ik} = 1 \qquad \forall\, i \in N$$

$$\sum_{i \in N} P_{ik} Y_{ik} \leq C_{max} \qquad \forall k \in M$$

$$Y_{ik} \in \{0,1\} \qquad \forall\, i \in N, k \in M$$

Apéndice C

A continuación, se muestran los modelos de programación lineal descritos en el capítulo 4. El modelo 1A es:

$$\text{Min } C_{max}$$

$$\sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}} X_{ijk} = 1 \qquad \forall\, k \in N$$

$$\sum_{i\in M}\sum_{\substack{k\in N\\ j\neq k}} X_{ijk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{\substack{h\in N_0\\ j\neq h\neq k}} X_{ihj} \geq X_{ijk} \qquad \forall\, i \in M,\, j \in N, k \in N, j \neq k$$

$$\sum_{k\in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j\in N_0\\ j\neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k\in N\\ j\neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}}\sum_{k\in N} s_{ijk}X_{ijk} \leq C_{max}$$

$$A_k + \sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}} (s_{ijk} + p_{ik})X_{ijk} \leq C_{max} \qquad \forall\, k\in N$$

$$A_k + \mu\left(1 - X_{ijk}\right) \geq A_j + p_{ij} + \sum_{\substack{h\in N_0\\ j\neq h\neq k}} s_{ihj}X_{ihj} \qquad \forall\, i\in M, j\in N, k\in N, j\neq k$$

$$A_k + \mu\left(1 - Z_{jk}\right) \geq A_j + \sum_{i\in M}\sum_{\substack{h\in N_0\\ j\neq h\neq k}} s_{ihj}X_{ihj} \qquad \forall\, j\in N, k\in N, j\neq k$$

El modelo 1B es:

$$\text{Min } C_{max}$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} = 1 \qquad \forall\, k \in N$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} X_{ijk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{\substack{h \in N_0 \\ j \neq h \neq k}} X_{ihj} \geq X_{ijk} \qquad \forall\, i \in M,\, j \in N, k \in N, j \neq k$$

$$\sum_{k \in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k \in N \\ j \neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} X_{ijk} \leq C_{max}$$

$$B_k + \sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} p_{ik} X_{ijk} \leq C_{max} \qquad \forall\, k \in N$$

$$B_k - s_{ijk} + \mu\left(1 - X_{ijk}\right) \geq B_j + p_{ij} \qquad \forall\, i \in M, j \in N, k \in N, j \neq k$$

$$B_k - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\, i \in M,\, k \in N$$

$$B_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihk} X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq B_j \qquad \forall\, j \in N, k \in N, j \neq k$$

El modelo 1C es:

$$\text{Min } C_{max}$$

$$\sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}} X_{ijk} = 1 \qquad \forall\, k \in N$$

$$\sum_{i\in M}\sum_{\substack{k\in N\\ j\neq k}} X_{ijk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{\substack{h\in N_0\\ j\neq h\neq k}} X_{ihj} \geq X_{ijk} \qquad \forall\, i \in M,\, j \in N, k \in N, j \neq k$$

$$\sum_{k\in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j\in N_0\\ j\neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k\in N\\ j\neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i\in M}\sum_{\substack{j\in N_0\\ j\neq k}}\sum_{k\in N} s_{ijk}X_{ijk} \leq C_{max}$$

$$C_k \leq C_{max} \qquad \forall\, k\in N$$

$$C_k - p_{ik} - s_{ijk} + \mu\left(1 - X_{ijk}\right) \geq C_j \qquad \forall\, i\in M, j\in N, k\in N, j\neq k$$

$$C_k - p_{ik} - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\, i\in M,\, k\in N$$

$$C_k - \sum_{i\in M}\sum_{\substack{h\in N_0\\ j\neq h\neq k}} (p_{ik} + s_{ihk})X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq C_j - \sum_{i\in M} p_{ij}X_{ijk} \qquad \forall\, j\in N, k\in N, j\neq k$$

El modelo 2A es:

$$\text{Min } C_{max}$$

$$Y_{ik} = \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ k \in N$$

$$Y_{ij} \geq \sum_{\substack{j \in N \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ j \in N$$

$$\sum_{i \in M} Y_{ik} = 1 \qquad \forall\, k \in N$$

$$\sum_{k \in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k \in N \\ j \neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} X_{ijk} \leq C_{max}$$

$$A_k + \sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} (s_{ijk} + p_{ik}) X_{ijk} \leq C_{max} \qquad \forall\ k \in N$$

$$A_k + \mu\left(1 - X_{ijk}\right) \geq A_j + p_{ij} + \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihj} X_{ihj} \qquad \forall\ i \in M, j \in N, k \in N, j \neq k$$

$$A_k + \mu\left(1 - Z_{jk}\right) \geq A_j + \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihj} X_{ihj} \qquad \forall\ j \in N, k \in N, j \neq k$$

El modelo 2B es:

$$\text{Min } C_{max}$$

$$Y_{ik} = \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ k \in N$$

$$Y_{ij} \geq \sum_{\substack{j \in N \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ j \in N$$

$$\sum_{i \in M} Y_{ik} = 1 \qquad \forall\, k \in N$$

$$\sum_{k \in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k \in N \\ j \neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} X_{ijk} \leq C_{max}$$

$$B_k + \sum_{i \in M} p_{ik} Y_{ik} \leq C_{max} \qquad \forall\ k \in N$$

$$B_k - s_{ijk} + \mu\left(1 - X_{ijk}\right) \geq B_j + p_{ij} \qquad \forall\ i \in M, j \in N, k \in N, j \neq k$$

$$B_k - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\ i \in M,\ k \in N$$

$$B_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} s_{ihk} X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq B_j \qquad \forall\ j \in N, k \in N, j \neq k$$

El modelo 2C es:

$$\text{Min } C_{max}$$

$$Y_{ik} = \sum_{\substack{j \in N_0 \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\ k \in N$$

$$Y_{ij} \geq \sum_{\substack{j \in N \\ j \neq k}} X_{ijk} \qquad \forall\, i \in M,\, j \in N$$

$$\sum_{i \in M} Y_{ik} = 1 \qquad \forall\, k \in N$$

$$\sum_{k \in N} X_{i0k} \leq 1 \qquad \forall\, i \in M$$

$$\sum_{\substack{j \in N_0 \\ j \neq k}} Z_{jk} = 1 \qquad \forall\, k \in N$$

$$\sum_{\substack{k \in N \\ j \neq k}} Z_{jk} \leq 1 \qquad \forall\, j \in N$$

$$\sum_{i \in M} \sum_{\substack{j \in N_0 \\ j \neq k}} \sum_{k \in N} s_{ijk} X_{ijk} \leq C_{max}$$

$$C_k \leq C_{max} \qquad \forall\, k \in N$$

$$C_k - p_{ik} - s_{ijk} + \mu\left(1 - X_{ijk}\right) \geq C_j \qquad \forall\, i \in M, j \in N, k \in N, j \neq k$$

$$C_k - p_{ik} - s_{i0k} + \mu(1 - X_{i0k}) \geq 0 \qquad \forall\, i \in M,\, k \in N$$

$$C_k - \sum_{i \in M} \sum_{\substack{h \in N_0 \\ j \neq h \neq k}} (p_{ik} + s_{ihk}) X_{ihk} + \mu\left(1 - Z_{jk}\right) \geq C_j - \sum_{i \in M} p_{ij} Y_{ij} \qquad \forall\, j \in N,\, k \in N,\, j \neq k$$

Apéndice D

			Mult					Pert				
n	m	s	Mín	Prom	Máx	Desvest	CV	Mín	Prom	Máx	Desvest	CV
20	2	[1,49]	385	402.3	415	6.1	1.5%	376	394	421	10.7	2.7%
		[1,99]	454	479.4	503	13.3	2.8%	442	467.4	515	19.6	4.2%
		[1,124]	442	457.7	480	11.3	2.5%	431	444.1	475	9.7	2.2%
	4	[1,49]	169	178.4	187	4.3	2.4%	168	178.2	199	8.3	4.7%
		[1,99]	258	293.6	315	10.2	3.5%	256	288.7	344	18.7	6.5%
		[1,124]	236	254.6	287	12	4.7%	226	251.7	316	21.8	8.7%
	6	[1,49]	128	137.2	145	4.3	3.1%	115	132	151	9.9	7.5%
		[1,99]	189	208.4	226	8.9	4.3%	180	194.8	231	15.7	8.1%
		[1,124]	229	241	258	8.6	3.6%	212	235.2	272	13.3	5.7%
	8	[1,49]	147	154	160	4.1	2.7%	133	149.7	168	7.1	4.7%
		[1,99]	153	170.9	183	8.1	4.7%	146	163.3	181	8.8	5.4%
		[1,124]	152	173.1	194	13.3	7.7%	141	159.1	195	15.5	9.7%
40	2	[1,49]	934	960.4	978	9.8	1.0%	907	932.3	954	12.4	1.3%
		[1,99]	975	1,000.10	1018	10	1.0%	921	950.2	1010	22.2	2.3%
		[1,124]	869	909.3	939	14.9	1.6%	814	853	935	27.9	3.3%
	4	[1,49]	365	384	407	9.8	2.6%	340	360.3	387	11.5	3.2%
		[1,99]	404	433.8	460	12.9	3.0%	343	389.4	452	25.5	6.5%
		[1,124]	496	541	576	20.3	3.8%	461	494.9	541	22.2	4.5%
	6	[1,49]	229	253.4	270	10.3	4.1%	204	219.3	250	11.2	5.1%
		[1,99]	344	373	399	13.4	3.6%	283	326.9	375	23.3	7.1%
		[1,124]	341	383.7	411	18.4	4.8%	289	345	402	24.4	7.1%
	8	[1,49]	202	234.2	249	9.5	4.1%	185	207.9	245	15.2	7.3%
		[1,99]	274	299.8	323	12.3	4.1%	234	261.9	283	13.3	5.1%
		[1,124]	308	345.2	364	12.4	3.6%	254	298.3	349	23.8	8.0%
60	2	[1,49]	1301	1,320.70	1340	10.4	0.8%	1254	1,283.20	1313	14.8	1.2%
		[1,99]	1309	1,344.40	1384	15.1	1.1%	1218	1,282.50	1322	26.2	2.0%
		[1,124]	1654	1,699.40	1740	21.1	1.2%	1516	1,620.60	1688	33.9	2.1%
	4	[1,49]	506	532.7	548	11.5	2.2%	472	507.9	540	18.3	3.6%
		[1,99]	664	689.7	723	14	2.0%	547	621.8	704	38.3	6.2%
		[1,124]	703	739.4	778	20.2	2.7%	550	646.6	726	47.9	7.4%
	6	[1,49]	359	385.3	415	11.8	3.1%	317	348.7	387	18.2	5.2%
		[1,99]	467	504.8	540	20.3	4.0%	390	439.1	510	28.7	6.5%
		[1,124]	543	577.7	615	18.8	3.3%	426	506.3	598	32.3	6.4%
	8	[1,49]	268	295.3	312	10.8	3.7%	235	262	296	13.5	5.2%
		[1,99]	389	428.9	457	17.9	4.2%	326	372.6	446	25.7	6.9%
		[1,124]	445	492.1	529	20.7	4.2%	354	419.6	479	36.3	8.7%
Promedio			480	507.8	531	12.5	2.5%	435	472.5	518	20.2	4.3%

Anexo 1: Portada de artículo publicado y datos de la revista

EURO Journal on Computational Optimization 10 (2022) 100034

Contents lists available at ScienceDirect

EURO Journal on Computational Optimization

journal homepage: www.elsevier.com/locate/ejco

A mixed integer formulation and an efficient metaheuristic for the unrelated parallel machine scheduling problem: Total tardiness minimization

Héctor G.-de-Alba [a], Samuel Nucamendi-Guillén [a,*], Oliver Avalos-Rosales [b]

[a] *Universidad Panamericana, Facultad de Ingeniería, Álvaro del Portillo 49, Ciudad Granja, Zapopan, 45010, Jalisco, Mexico*
[b] *Centro de Investigación en Matemáticas Aplicadas, Universidad Autónoma de Coahuila, Unidad Camporredondo s/n, Edificio S, Saltillo, 25020, Coahuila, Mexico*

ARTICLE INFO

Keywords:
Total tardiness
Unrelated parallel machines
Scheduling
Mixed integer programming
Iterated local search algorithm

ABSTRACT

In this paper, the unrelated parallel machine scheduling problem with the objective of minimizing the total tardiness is addressed. For such a problem, a mixed-integer linear programming (MILP) formulation, that considers assignment and positional variables, is presented. In addition, an iterated local search (ILS) algorithm that produces high-quality solutions in reasonable times is proposed for large size instances. The ILS robustness was determined by comparing its performance with the results provided by the MILP. The instances used in this paper were constructed under a new approach which results in tighter due dates than the previous generation method for this problem. The proposed MILP formulation was able to solve instances of up to 150 jobs and 20 machines. Regarding the ILS, it yielded high-quality solutions in a reasonable time, solving instances of a size up to 400 jobs and 20 machines. Experimental results confirm that both approaches are efficient and promising.

* Corresponding author.
E-mail address: snucamendi@up.edu.mx (S. Nucamendi-Guillén).

https://doi.org/10.1016/j.ejco.2022.100034

Documento enviado a revisión el 07/12/2021, revisado el 04/05/2021, aceptado el 23/06/2022, y publicado el 12/07/2022

JCR YEAR

2021

EURO Journal on Computational Optimization

Open Access since 2013

ISSN

2192-4406

EISSN

2192-4414

JCR ABBREVIATION

EURO J COMPUT OPTIM

ISO ABBREVIATION

EURO J. Comput. Optim.

Journal information

EDITION

Emerging Sources Citation Index (ESCI)

CATEGORY

OPERATIONS RESEARCH & MANAGEMENT SCIENCE - ESCI

LANGUAGES	REGION	1ST ELECTRONIC JCR YEAR
English	NETHERLANDS	2020

Publisher information

PUBLISHER	ADDRESS	PUBLICATION FREQUENCY
ELSEVIER	RADARWEG 29, 1043 NX AMSTERDAM, NETHERLANDS	4 issues/year

Journal's performance

Journal Citation Indicator (JCI)

Export

0.34

The Journal Citation Indicator (JCI) is the average Category Normalized Citation Impact (CNCI) of citable items (articles & reviews) published by a journal over a recent three year period. The average JCI in a category is 1. Journals with a JCI of 1.5 have 50% more citation impact than the average in that category. It may be used alongside other metrics to help you evaluate journals.

Learn more

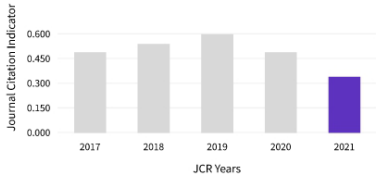
Total Citations

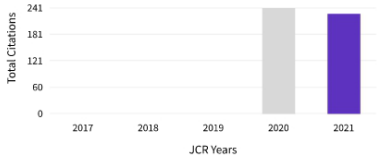
Export

228

The total number of times that a journal has been cited by all journals included in the database in the JCR year. Citations to journals listed in JCR are compiled annually from the JCR years combined database, regardless of which JCR edition lists the journal.

14





View all years

Open Access (OA)

Export

The data included in this tile summarizes the items published in the journal in the JCR data year and in the previous two years. For example, in the 2020 JCR data, released in June 2021, the Open Access (OA) data show the publication model (Gold OA or subscription) of materials published in 2018, 2019 and 2020, and citations in 2020 to these items. This three-year set of published items is used to provide descriptive analysis of the content and community of the journal. Learn more

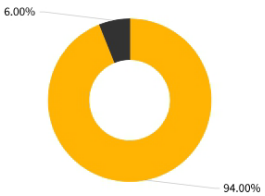
Items

TOTAL CITABLE	% OF CITABLE OA
47	100.00%

CITABLE

GOLD OPEN ACCESS
47 / 94.00%

SUBSCRIPTION AND FREE TO READ
0 / 0.00%



NON-CITABLE

OTHER (NON CITABLE ITEMS)
3 / 6.00%

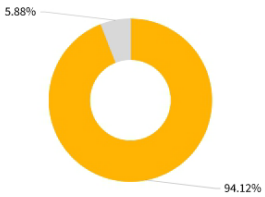
Citations*

TOTAL CITABLE	% OF CITABLE OA
32	100.00%

CITABLE

GOLD OPEN ACCESS
32 / 94.12%

SUBSCRIPTION AND FREE TO READ
0 / 0.00%



NON-CITABLE

OTHER (NON CITABLE ITEMS)
0 / 0.00%

UNLINKED CITATIONS
2 / 5.88%

*Citations in 2021 to items published in [2019 - 2021]

Rank by Journal Citation Indicator (JCI)

Journals within a category are sorted in descending order by Journal Citation Indicator (JCI) resulting in the Category Ranking below. A separate rank is shown for each category in which the journal is listed in JCR. Data for the most recent year is presented at the top of the list, with other years shown in reverse chronological order. Learn more

CATEGORY

OPERATIONS RESEARCH & MANAGEMENT SCIENCE

81/100

JCR YEAR	JCI RANK	JCI QUARTILE	JCI PERCENTILE
2021	81/100	Q4	19.50
2020	62/99	Q3	37.88
2019	57/99	Q3	42.93
2018	59/98	Q3	40.31
2017	64/96	Q3	33.85

Citation network

Cited Half-life

4.8 years

The Cited Half-Life is the median age of the items in this journal that were cited in the JCR year. Half of a journal's cited items were published more recently than the cited half-life.

TOTAL NUMBER OF CITES

228

NON SELF-CITATIONS

226

SELF-CITATIONS

2

Citing Half-life

10.3 years

The Citing Half-Life is the median age of items in other publications cited by this journal in the JCR year.

TOTAL NUMBER OF CITES

1,667

NON SELF-CITATIONS

1,665

SELF-CITATIONS

2

Cited Half-life Data | Citing Half-life Data

Export

Anexo 2: Carta de completación de estancia

CENTRO DE INVESTIGACIÓN EN MATEMÁTICAS APLICADAS
UNIVERSIDAD AUTÓNOMA DE COAHUILA

Saltillo Coahuila a 25 de Enero del 2020

Estancia de Investigación de M.I Héctor R. García de Alba V.

Por medio de la presente hago constar que el M.I. Héctor R. García de Alba V., doctorando del programa de Doctorado en Ingeniería de la Facultad de Ingeniería de la Universidad Panamericana, campus Guadalajara, ha concluido su estancia de investigación académica en el Centro de Investigación en Matemáticas Aplicadas (CIMA) de la UAdeC, bajo la supervisión del Dr. Oliver Ávalos Rosales, profesor investigador de dicho centro.

Dicha estancia tuvo una duración de **60 horas con docente y de 60 horas independientes**.

El objetivo de dicha estancia fue:

Desarrollar modelos matemáticos y algoritmos para resolver el problema de Secuenciación de Tareas en Máquinas Paralelas no Relacionadas con Recursos Compartidos y Tiempos de Preparación Dependientes de la Secuencia.

Los resultados de la estancia fueron:

1. Varias formulaciones para el problema estudiado.
2. Experimentación computacional para determinar el alcance de los modelos
3. El diseño de algoritmos heurísticos que formaran parte del algoritmo metaheuristico de su trabajo de tesis

Sin más por el momento quedo atento a cualquier aclaración.

Atentamente

Oliver Avalos R.

Dr. Oliver Avalos Rosales
Profesor-Investigador
CIMA-UAdeC
o.avalos@uadec.edu.mx

Anexo 3: Certificado de participación en ELAVIO 2022