UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

TÉCNICAS DE OPTIMIZACIÓN PARA RESOLVER UN PROBLEMA DE PROGRAMACIÓN DE PRODUCCIÓN DE UNA LÍNEA DE ENSAMBLE

POR

RAFAEL MUÑOZ SÁNCHEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE DOCTOR EN INGENIERÍA CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

MARZO, 2018

UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO

TÉCNICAS DE OPTIMIZACIÓN PARA RESOLVER UN PROBLEMA DE PROGRAMACIÓN DE PRODUCCIÓN DE UNA LÍNEA DE ENSAMBLE

POR

RAFAEL MUÑOZ SÁNCHEZ

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE DOCTOR EN INGENIERÍA CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

MARZO, 2018

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Técnicas de optimización para resolver un problema de programación de producción de una línea de ensamble», realizada por el alumno Rafael Muñoz Sánchez, con número de matrícula 1467571, sea aceptada para su defensa como requisito parcial para obtener el grado de Doctorado en Ingeniería con Especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dr. Iris Abril Martínez Salazar
Director

Dr. José Luis González Velarde
Revisor

Dr. Yasmin Ríos Solis
Revisor

Dr. Francisco Román Angel Bello Acosta
Revisor

Dr. Igor Litvinchev Semionovich
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirección de Estudios de Posgrado

SUBDIRECCIÓN DE ESTUDIOS DE POSGRADO FIME U.A.N.L.

San Nicolás de los Garza, Nuevo León, diciembre 2017

A mi esposa Quetzali, mi hijo Diego y mis padres Ramon y Silvia

"Si lo puedes soñar, lo puedes lograr." Walt Disney.

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Quiero agradecer a Dios por poder concluir un objetivo más dentro de mi vida y haberme permitido compartirlo con mi familia, amigos y maestros.

En primer lugar, agradezco a la Dra. Iris Abril por todas sus enseñanzas, por su tiempo, su dedicación y su apoyo incondicional durante mi estancia en el doctorado. Estos años me permitieron conocer su brillante inteligencia y su enorme calidad de persona, mi más sincera y profunda admiración para una gran persona, en el ámbito profesional y como ser humano. Gracias por ser mi asesor, por creer y confiar en mí. De corazón muchas gracias Dra. Iris.

Agradezco a la Dra. Yasmín Ríos por todos sus grandes consejos, es un honor haber participado en sus clases. Considero que sus aportes enriquecen mucho las aplicaciones reales dentro del scheduling. Gracias por su tiempo y dedicación hacia mis preguntas, y sobre todo por motivarme al área de las aplicaciones. Admiro y respeto mucho su trabajo. Gracias Dra. Yasmín.

Al Dr. José Luis, agradezco sus consejos durante este trabajo y los que hemos compartido. Usted ha sido y será un icono para muchos dentro de la investigación de operaciones, no solamente en México. Respeto y admiración a todos sus aportes que ha compartido a lo largo de su trayectoria. Para muchos, el mejor investigador que hay en México.

Al Dr. Francisco y al Dr. Igor agradezco su tiempo y dedicación para la revisión de esta tesis y por los comentarios y sugerencias que fueron de gran relevancia para

la redacción del mismo.

Agradezco a mi esposa Quetzali por acompañarme durante todo este proceso, por su apoyo incondicional, su equilibrio, su amor y entrega que brinda a nuestra familia. A mi hijo Diego, por sus sonrisas, juegos y amor que nos brinda. A mis padres que me brindaron gran parte de mi educación y formación. A mi hermana por estar presente en gran parte del camino. Gracias familia.

Agradezco a mis amigos Pedro, Pamela, Álvaro, Diego, Jobish y demás por brindarme su amistad durante este tiempo.

Por último, agradezco al CONACYT por el apoyo de manutención brindada, así como a la FIME y a la UANL por permitirme estudiar en tan prestigiadas instituciones.

Resumen

Rafael Muñoz Sánchez.

Candidato para obtener el grado de Doctorado en Ingeniería con Especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Técnicas de optimización para resolver un problema de programación de producción de una línea de ensamble..

Número de páginas: 85.

Objetivos y método de estudio: El objetivo del presente trabajo es el estudio de un problema de programa de producción de una empresa del sector automotriz. La empresa requiere mejorar la programación de la producción de dicha línea de producción, pues la considera crítica dentro de su proceso productivo. Dadas las características de la línea de producción, se puede clasificar el problema como un caso particular de un Hybrid Flow Shop Scheduling (HFS).

Una de los aspectos relevantes de la línea de producción es que, en una de las últimas etapas del proceso, se realiza una pieza mediante el ensamblado de dos componentes procesados en la misma línea. En la línea de producción, se fabrican una familia de piezas con diferentes tiempos de procesamiento y tiempos de preparación

de las máquinas debido a programación de las mismas y cambio de herramientas. Dada ciertas órdenes de piezas, se busca minimizar el tiempo total de ciclo requerido para producir todas ellas. La línea de producción bajo análisis cuenta con más de 15 etapas dentro del proceso de producción, cada una de ellas con un conjunto de máquinas en paralelo.

En este trabajo se presentan diferentes alternativas de solución para resolver el problema. El primer enfoque consiste en proponer un modelo matemático que representa la situación actual de la empresa. Debido a la complejidad del problema, se proponen tres métodos heurísticos para la solución de este. El primer método se basa en la metodología GRASP, mientras que el segundo se presenta una adaptación de la metaheurística BRKGA. Por último, se propone un algoritmo mathheuristic, el cual aprovecha la estructura presentada del modelo matemático.

Contribuciones y conlusiones: Se presenta un nuevo modelo matemático en el area de problemas de schedulig. Se proponen dos nuevas metodologías heurísticas en problemas de scheduling y una adaptación de la metaheurística BRKGA. De acuerdo a las comparación entre los algoritmos propuestos, el problema es resuelto en un tiempo razonable y con soluciones de buena calidad.

Firma del asesor: 

Dr. Iris Abril Martínez Salazar

Capítulo 1

Introducción

Este capítulo está organizado de la siguiente manera: en la sección 1.1, se presenta la motivación del estudio de esta investigación. En la sección 1.2, se describe la problemática que aborda este trabajo. En la sección 1.3, se muestra la justificación de esta investigación. En las secciones 1.4 y 1.5, se plantean las hipótesis y objetivos de la investigación, respectivamente. Por último, en la sección 1.6 se muestra la contribución científica y en la sección 1.7, se indica la estructura del contenido de la tesis.

1.1 Motivación

Existen diferentes niveles de decisión en las empresas del sector industrial. Estos niveles pueden ser clasificados como: nivel estratégico (nivel alto), nivel táctico (nivel medio) y nivel operativo (nivel bajo). En cada uno de estos niveles se involucran decisiones de suma importancia. La programación de producción es una actividad que se encuentra en el nivel operativo. En los sistemas de manufactura, esta actividad es alimentada de un plan maestro de producción, de requerimientos de material y de la capacidad de plantas. Esto se puede ver en la figura 1.1, un diagrama de flujo de un sistema de manufactura presentado en Pinedo [42]. Como puede verse en este diagrama, existe una interacción con la planeación y la ejecución en planta, siendo

factible algunos cambios de la programación inicial debido a diferentes causas que pueden suceder en piso.

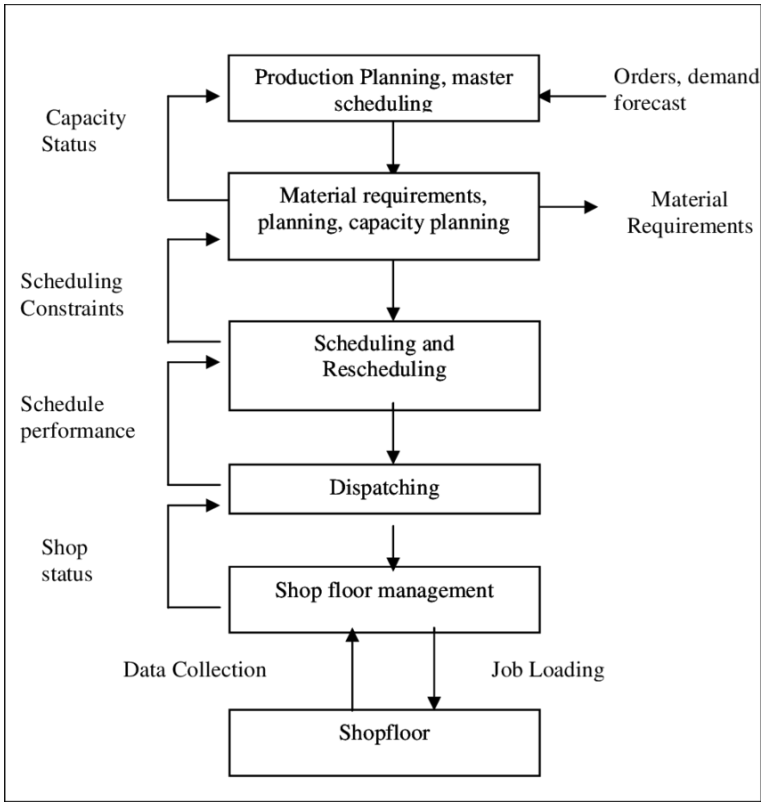


Figura 1.1: Diagrama de flujo de sistemas de manufactura

En la actualidad, la mayor parte de las empresas realizan la programación de producción de forma manual. Es decir, sus sistemas trabajan en base a decisiones tomadas por un programador de producción. Debido a la inmensa cantidad de información que interviene, como lo es, la cantidad de máquinas, la capacidad de cada una de ellas, la cantidad de productos, los componentes de cada productos, el total de trabajos de cada producto a procesar, etc., tiende a ser un problema muy complejo para cualquier persona.

Debido a la complejidad de esta tarea, es necesario que las empresas tomen decisiones basadas en modelos matemáticos y algoritmos que beneficien la optimización de sus sistemas de producción. Una decisión basada en estas herramientas puede minimizar el tiempo total de producción, mejorar los tiempos perdidos en las

máquinas, mejorar las fechas de entregas el cliente, entre otras cosas. Lo anterior, no solo lleva un beneficio interno en la empresa, sino también, a un buen servicio al cliente.

La programación de producción o scheduling, como se le conoce en la literatura, es uno de los problemas con mayor complejidad dentro de la optimización discreta. En la literatura existen diversos trabajos relacionados con el scheduling en diferentes industrias. El trabajo presentado por P. Gomez-Gasquet [20] relaciona un problema de secuenciación de operaciones en un sistema hibrido (HFS) en la industria de la cerámica. Por otra parte, el trabajo presentado por Behnamian y Zandieh [7] relaciona una aplicación del scheduling en la industria de los semicondutores. Por último, la industria del acero es otro caso donde se presenta este tipo de problemas, el trabajo mostrado por Liao et al. [38] muestra una aplicación real en esta industria.

Este trabajo es motivado de una aplicación real de una línea de producción de ensamble de una empresa del sector automotriz. La compañía se encuentra ubicada en Escobedo Nuevo León. Esta empresa cuenta con varias líneas de producción, sin embargo, este trabajo se concentra en la programación de producción de la línea que es considerada por parte de la empresa, como el proceso más crítico dentro todo su sistema de producción. Se propone una herramienta basada en un modelo matemático y en metodologías heurísticas para la solución del problema.

1.2 Descripción del problema

La línea de producción a analizar es divida en dos sub-líneas de producción paralelas, las cuales son unidas en una etapa para ensamblar el producto. Cada sub-línea produce una componente. La sub-línea uno produce la corona, la cual llamaremos componente uno, y la sub-línea dos produce el piñón, el cual llamaremos componente dos. Existen diferentes modelos de los componentes uno y dos, a estos modelos los llamaremos productos. Se conoce a priori la demanda de cada producto.

En cada sub-línea se cuenta con un conjunto de etapas, y en cada una de estas se tiene un conjunto de máquinas en paralelo.

La figura 1.2 representa la línea de producción del caso real de la empresa. Cada cuadro representa una etapa del proceso, y en cada uno de estos existen máquinas paralelas. Se puede ver en color azul celeste la sub-línea de producción de la componente uno, así como, en color lila la sub-línea de producción de la componente dos. Como se puede ver ambas son paralelas y coinciden en la etapa de color amarillo, la cual representa la etapa de ensamble. Además, se sabe que una sub-línea de producción es hasta tres veces más rápida que la otra.

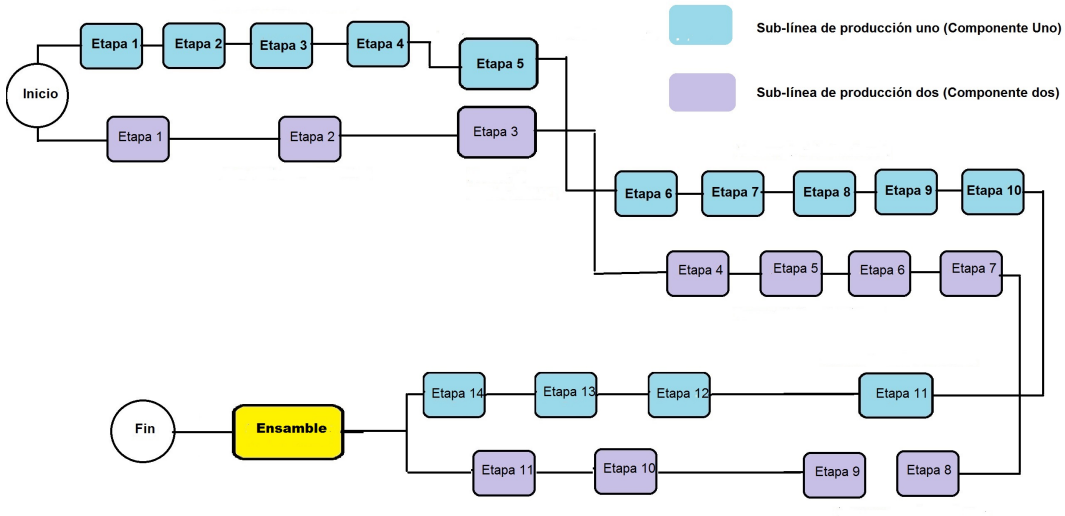


Figura 1.2: Línea de producción

El problema puede ser considerado como un caso especial de un Hybrid Flow Shop (HFS). Ruiz et al. [46] describe a un HFS como un conjunto de n trabajos que tienen que ser procesados en serie por un conjunto de m etapas, en las cuales se tienen en cada una de estas k máquinas paralelas. En la literatura es conocida a la etapa de ensamble como aquella en la cual los dos componentes son unidos para formar un solo producto. Así, cada sub-línea de producción puede verse como un HFS, donde existe una etapa de ensamble en la que se juntan ambas sub-líneas. En el capítulo dos se detalla el problema de HFS.

1.3 Justificación

El impacto de esta tesis se puede clasificar en teórico y práctico. En el contexto práctico, la competencia actual en el sector industrial lleva a la necesidad por parte de las empresas de adquirir tecnología que implique mejoras a sus procesos operativos. En este caso, la empresa dedicada a ensamblar partes de automóviles, decidió optar por una herramienta que mejore su programación de producción dentro de su línea de ensamble.

El desarrollo final de esta tesis, es un modelo matemático y un conjunto de metodologías heurísticas que se pueden utilizar para la programación de producción de la línea de ensamble de la empresa. La integración de estas metodologías con la colaboración del factor humano (programador de producción) ayudará a mejorar el sistema de producción. Esto conducirá a una reducción del tiempo total de flujo del proceso y ayudará a disminuir los tiempos muertos en las máquinas. Favoreciendo el crecimiento económico de la empresa y la satisfacción de los clientes.

Por otra parte, en el contexto teórico, estos estudios ayudarán a complementar el conjunto de conocimientos de la programación de producción en líneas de ensamble. Además, esta investigación contribuirá a incrementar el estado del arte de los siguientes temas: Adaptación de algoritmos metaheurísticos en sistemas de producción, variantes de modelación para el problema de HFS e implementación de una heurística basada en programación matemática.

1.4 Hipótesis

Desarrollo de una formulación matemática permite representar la línea de producción aquí estudiada.

La implementación de dos procedimientos metaheurísticos basados en GRASP y BRKGA perimiten obtener soluciones de calidad para el problema de este trabajo.

La integración de la programación matemática y técnicas heuristicas permiten obtener soluciones de calidad para el problema estudiado.

1.5 Objetivos

El objetivo principal de este trabajo consiste en estudiar, proponer y comparar diferentes alternativas de solución para resolver el problema de programación de producción de una línea de ensamble. Los métodos de solución están basados en programación matemática, técnicas metaheurísticas y matheuristics.

Con el fin de lograr el objetivo principal se puntualizan los siguientes objetivos particulares:

- Formulación matemática del problema con un modelo de optimización discreta, para abordar el problema de programación de producción de una línea de ensamble.
- Adaptación e implementación de la metaheurística GRASP para encontrar soluciones de buena calidad.
- Implementación de la metaheurística BRKGA para encontrar buenas soluciones.
- Diseño e implementación de una técnica basada en una metaheurística basada

en programación matemática.

- Afinación de parámetros de las diferentes técnicas de solución.
- Comparación y análisis de resultados de las diferentes metodologías de solución.

1.6 Contribución científica

Después de una extensa revisión de literatura, nos dimos cuenta que el HFS con las características que se proponen en este trabajo, no ha sido abordado en la literatura por otras autores. Se decidió clasificar la contribución científica en dos aspectos, la modelación y la metodología de solución:

Modelación

El modelo matemático propuesto, decide la programación de producción de la línea de ensamble mediante variables de asignación, secuenciación y tiempo. Minimiza el tiempo total de ciclo de todas las tareas a realizar, aunado a la llegada simultánea de las dos componentes de cada producto a la etapa de ensamble. El modelo contempla incompatibilidad producto-máquina.

Metodología de solución

Se realizó la adaptación, implementación y validación de las metodologías GRASP y BRKGA. Se obtuvieron soluciones de buena calidad al problema de HFS de una línea de ensamble.

En la metodología basada en Matheurístic, se consideró la línea de producción en sentido inverso. Es decir, se realiza en primer orden la programación de la etapa de ensamble continuando con las demás etapas y terminando en la fase inicial. En esta metodología se propuso una técnica basada en matheuristics, descomponiendo el problema en dos sub-problemas, el primero de asignación y el segundo de secuenciación.

1.7 Estructura de la tesis

Este documento está estructurado en seis capítulos. Este capítulo corresponde a la presentación del tema de esta investigación. Se muestra la motivación por la cual se resuelve este problema, se describe el problema, las hipótesis y los objetivos de esta tesis, por último, se muestra la contribución científica.

En el capítulo dos se describirá el marco teórico, así como los conceptos necesarios para el estudio de este problema. Se describen las características de los sistemas de producción basados en un Hybrid Flow Shop. Además, se describen algunas variantes de solución que han sido abordados en la literatura. Así como se presenta la relación de este problema con otros modelos en la literatura.

En el capítulo tres se describe a detalle el planteamiento del problema. Se presenta la formulación matemática del problema. Se describen todos los elementos, como las variables, restricciones y función objetivo. Se muestra la complejidad del problema y la justificación de optar por otras técnicas de solución.

En el capítulo cuatro se muestra a detalle cada una de las metodologías de solución. Se presenta la metodología GRASP, sus pseudocódigos y se detallan las características de su implementación. Así como se muestra cada una de las etapas del BRKGA implementado. Se detalla la etapa de descodificación. Por último, se justifica el análisis de la línea de producción en sentido inverso y posteriormente se describe el algoritmo basado en matheuristic.

En el capítulo cinco se describe el ambiente computacional y se presentan los resultados computacionales de las metodologías propuestas. Por último, se comparan las técnicas de solución en la calidad de la función objetivo y tiempo. En el último capítulo se presentan las conclusiones y la propuesta de trabajo a futuro.

Capítulo 2

Estado del arte

En el presente capítulo se aborda la programación de producción y sus diferentes ambientes en el sector productivo, así como definiciones y nomenclaturas dentro de la literatura. Después, se realiza una extensa revisión de literatura de los trabajos relacionados y las metodologías de solución empleadas.

2.1 Introducción

Dentro de los sistemas de producción existen diferentes niveles en la jerarquía de las decisiones, estos son: el nivel estratégico (primer nivel), el nivel táctico (segundo nivel) y el nivel operativo (el tercer nivel). Lo anterior podemos verlo en Domínguez Machuca et al. [13]. En el trabajo de Gaither et al. [18], se considera al primer nivel de producción como aquel que establece la estrategia de producción, que considera los criterios por los cuales será invertido el esfuerzo del sistema de producción a través de decisiones estratégicas en un largo plazo.

Por otra parte, en el trabajo Domínguez-Machuca et al. [13], se define el nivel táctico como aquel que se encarga de definir las necesidades de entregar el volumen de producción dada una capacidad estimada en cada periodo. A este término se le puede conocer como planeación agregada.

En Pinedo [41], se define al nivel operativo como el proceso en cual se asignan los recursos en un corto plazo con el fin de dar cumplimiento a los programas de producción. En este trabajo seguimos la definición propuesta por Alharkan [3], el cual define la programación de producción (scheduling) como: asignar a cada operación de cada trabajo un tiempo de inicio y fin dentro de una máquina, considerando las relaciones de precedencia. Además, establece la diferencia entre los términos de scheduling y secuenciación. Este último término indica que en cada máquina se tiene que establecer un orden en que los trabajos serán procesados por ésta. Como se puede apreciar, en la secuenciación no implica la decisión de los tiempos de inicio y fin.

En la actualidad, los problemas de secuenciación y programación cuentan con diversas aplicaciones reales en diferentes industrias, acontinuación se presentan algunos casos: En la industria del transporte, el trabajo presentado por Ibarra-Rojas y Ríos-Solís [32], en el que se se propone un modelo matemático y un algoritmo VNS para determinar los tiempos de salida de un conjunto de viajes, buscando maximizar el número de sincronizaciones de las líneas de autobuses este caso en la ciudad de Monterrey.

En hospitales, al diseñar la programación de los médicos en salas de emergencia, considerando todas los requisitos como turnos, reglas de antigüedad, periodos vacacionales, preferencias, entre otros. Un trabajo que aborda esta problematica es el presentado por Beaulieu et al. [6], en el cual proponen un modelo matemático para un hospital en Montreal.

En procesos de manufactura, la tesis presentado por Herrera [30], propone un modelo matemático y una heurística basada en GRASP para resolver la programación de una línea producción que fabrica partes de automóviles.

2.2 Nomenclatura del scheduling

De acuerdo a la sección anterior, en este trabajo se define la programación de producción o scheduling, como asignar a cada operación de cada trabajo un tiempo de inicio y fin dentro de una máquina, considerando las relaciones de precedencia. A continuación se presenta la nomenclatura de estos problemas.

La notación para la clasificación de los problemas de scheduling fue introducida por Graham et al. [24] Esta notación clasifica en tres campos $\alpha|\beta|\gamma$, donde α especifica el ambiente de la máquina, β representa las características de las tareas y γ presenta el criterio de optimalidad.

Enseguida describimos los posibles ambientes de máquinas para el parámetro α:

Sistema de una sola máquina (l). Sólo existe una máquina, por lo cual, es el ambiente más simple. Es considerado el caso especial de todos los demás ambientes.

En los ambientes de máquinas en paralelo, todos cuentan con m máquinas en paralelo y cualesquier trabajo puede ser procesado en cualequier máquina. Las diferencias son las siguientes:

Máquinas idénticas en paralelo (P_m). Sea $p_{i,j}$ el tiempo de procesamiento del trabajo j en la máquina i, en cada máquina el mismo tiempo, esto se puede denotar como: $p_{i,j} = p_j$.

Máquinas paralelas uniformes. Cada máquina i cuenta con una velocidad v_i diferente a otras las máquinas, sin embargo, dentro de cada máquina se tiene la misma velocidad para todos los trabajos. Por lo cual, el tiempo de procesamiento del trabajo j en la máquina i es igual p_j/v_i.

Máquinas paralelas no relacionadas (R_m). En este ambiente la velocidad de cada máquina i depende de cada trabajo j. Así, el tiempo de procesamiento del

trabajo j en la máquina i es igual $p_j/v_{i,j}$

Flow Shop (P_m). Existen m máquinas en serie y cada trabajo debe ser procesado en cada una de ellas siguiendo el mismo flujo, es decir, se procesan en la máquina uno, después en la máquina dos y así sucesivamente hasta terminar.

Hybrid Flow Shop (FF_c). Es la generalización del flow shop. Ahora se tienen k etapas en serie y en cada una de estas se tienen máquinas en paralelo, no necesariamente la misma cantidad de máquinas en cada etapa. Cada trabajo tiene que pasar por todas y cada una de las etapas en el mismo flujo. También es conocido como Flexible Flow Shop o Multiprocess Flow Shop. Más adelante se hablará a detalle.

Job Shop (J_m). Sean m máquinas y n trabajos, entonces, cada trabajo tiene una ruta predefinida. Existen dos posibles casos, uno es cuando cada trabajo solo visita a lo más una vez cada máquina y el segundo, cuando los trabajos pueden visitar más de una vez cada una de ellas.

Hybrid Job Shop (FJ_c). Es la generalización del job shop con ambiente máquinas en paralelo. Se tienen estaciones de trabajo y en cada uno se cuenta con máquinas idénticas en paralelo. Cada trabajo tiene su propia ruta a seguir.

A continuación se muestran algunas de las restricciones que son entradas del parámetro β:

Tiempos de liberación (r_j).Si este parámetro aparece en el campo, entonces el trabajo j no puede empezar su proceso antes de su fecha de lanzamiento r_j. Sino aparece este parámetro, entonces, el trabajo puede empezar en cualquier tiempo.

Tiempo de preparación ($s_{j,k}$). Los $s_{j,k}$ representan los tiempos de preparación que dependen de la secuencia de los trabajos, en este caso, primero es el trabajo j y después el k.

Preemptions ($prmp$). Si existe esta condición, los trabajos no podrán ser

interrumpidos una vez que inicien en una máquina.

Precedencia (*prec*). Esta restricción implica que un trabajo depende de la terminación de otro.

Breakdowns (*brkdwn*). Implica que las máquinas no están disponibles de forma continua.

Recirculación (*recrc*).Este ambiente implica que un trabajo visita a una máquina más de una ocasión.

Permutation (*prmu*). El orden de los trabajos antes de ser procesados en una máquina es de acuerdo con la regla FIFO.

Por último, se describen algunos de los criterios de optimalidad del parámetro γ. En todos ellos se implica la minimización:

Makespan (C_{mak}). Sea C_j el tiempo de completamiento del trabajo j, esto es, el tiempo final al salir del sistema del trabajo j, el makespan es igual el máximo tiempo de completamiento de todas las tareas.

Para los siguientes criterios se definen previamente la fecha de entrega, el retraso y la tardanza. La fecha de entrega se refiere a la fecha especificada de cuándo debe estar listo un trabajo. Mientras que el retraso L_j, se define como la diferencia entre el tiempo de completamiento del trabajo menos la fecha de entrega $L_j = C_j - d_j$. Por otra parte, la tardanza T_j difiere al retraso en que si la diferencia es negativa entre el tiempo de completamiento y la fecha de entrega, entonces su valor es cero, por lo cual, solo se considera la demora en el tiempo de entrega $T_j = max\{C_j - d_j, 0\}$.

Máximo retraso(L_{mak}). Se define L_{max} como el máximo de los retrasos de cada trabajo L_j. Esta medida implica la violación de restricciones de fechas de entrega.

Tiempo total ponderado de completamiento ($\sum w_j * C_j$). Minimizar el tiempo total ponderado de completamiento implica minimizar los costos de inventa-

rio o almacenamiento.

Tiempo total ponderado de tardanza ($\sum w_j * T_j$). Minimizar el tiempo total ponderado de tardanza implica la minimización de costos.

2.3 El problema del Hybrid Flow Shop

El problema de programación del flow shop ha sido tratado desde la década de los cincuenta. El trabajo seminal presentado en Johnson [35] muestra el clásico problema con un conjunto de trabajos que tienen que ser procesados en un conjunto de etapas en serie, en las cuales existe una sola máquina en cada etapa. A lo largo de la historia, la producción industrial ha crecido enormemente, por este motivo las empresas tienen la necesidad de incrementar sus capacidades. Una alternativa para aumentar la capacidad es el aumento de máquinas dentro de un mismo proceso. Es aquí donde nace el Hybrid Flow Shop.

Como se mencionó en el capítulo uno, en este trabajo se consideran las características de un Hybrid Flow Shop (por sus siglas en ingles HFS) presentadas por Ruiz y Vázquez-Rodríguez [46] en un ambiente de manufactura.

Se tiene un conjunto de N trabajos, los cuales tienen que ser procesados en serie por un conjunto de K etapas para optimizar una función objetivo. Usualmente, los problemas poseen las siguientes propiedades:

- El número de etapas en la línea de producción es de al menos dos
- Cada etapa k tiene al menos una máquina M_k y al menos una etapa tiene al menos dos máquinas en paralelo.
- Todos los trabajos tienen que ser procesados en el mismo flujo, etapa 1, etapa 2, ..., etapa $|K|$. En ocasiones un trabajo puede saltarse algún conjunto de etapas.

- Cada trabajo i en la etapa k requiere un tiempo de procesamiento $p_{i,k}$.

De acuerdo a la revisión de literatura realizada, podemos decir que los problemas de HFS pueden clasificarse en base a la dirección de sus flujos de producción. Los primeros son considerados como flujos unidireccionales, en estos los trabajos empiezan en la primer etapa y termina en la última. Este es el caso de nuestro problema, el cual se describirá más adelante. Mientras que el segundo caso, son caracterizados por el flujo reentrantes, aquí los trabajos pueden entrar más de una vez a cada etapa, como en el trabajo de Choi et al. [12].

Como se pudo observar en la sección anterior, existen diversos objetivos dentro de los problemas de secuenciación. En la actualidad, diversas compañías prefieren minimizar los tiempos totales de ciclo, y a su vez, benefician la utilización de sus máquinas o estaciones de trabajo. Asociados a estos objetivos se encuentra el Makespan. Este objetivo es el más común dentro de la literatura. En la siguiente sub-sección se describen algunos trabajos relacionados con el objetivo del Makespan y otros. Posteriormente, se presentan los trabajos más destacados del HFS en la industria.

2.3.1 El Makespan y otros criterios en el HFS

Existen diversos trabajos relacionados con el criterio del makespan y los problemas del HFS y sus casos paticulares. En la mayoría de estos trabajos usan estrategias de solución como técnicas heurísticas. En el trabajo de Guinet y Solomon [25], usan algoritmos heurísticos para resolver un HFS que tiene como objetivos la minimización del makespan y la tardanza. Otro trabajo es el propuesto por Haouari et al. [28], en este trabajo resuelven un HFS con dos etapas con un algoritmo de ramificación y acotamiento de forma óptima, minimizando el tiempo total de completamiento, aunque en la sección experimental se observa que no resuelve en optimalidad todas las instancias medianas (20-50 trabajos). Mientras que en Rebaine [43], se construyen

dos algoritmos heurísticos para un flexible flow shop en el cual se busca minimizar el objetivo del makespan.

En algunos de esta revisión de literatura algunos trabajos utilizan el algoritmo de Johnson, por lo cual se explican a continuación.

Este algoritmo se usa cuando se tienen dos máquinas en serie. Primero se decide la secuenciación de la primer máquina y después se repite esta secuencia en la otra máquina. Al inicio se enlistan los trabajos en cada máquina con sus respectivos tiempos de procesamiento. Después, se selecciona el trabajo con menor tiempo de procesamiento, si este tiempo pertenece a la primer máquina, este trabajo se ordena en la posición inicial, de lo contrario, si pertenece a otra máquina se debe secuenciar en la posición final de la primer máquina. Después se selecciona el siguiente trabajo con el menor tiempo y se decide secuenciar en la segunda posición o en la penúltima, dependiendo si su tiempo pertenece a la primer máquina o a la segunda. Así hasta terminar con todos los trabajos. Como se puede apreciar, el algoritmo decide secuenciar al inicio o al final de la máquina.

Dos trabajos interesantes son los propuestos por Kurz et al [36] y Logendran et al. [40] en los cuales se estudia un HFS con tiempos de preparación dependientes de la secuencia. En la primer investigación se proponen tres heurísticas basadas en inserción y en adaptaciones del algoritmo de Johnson. Mientras que en la segunda investigación consideran algoritmos basados en la Metaheurística tabu search, estos algoritmos varían en la solución inicial. Como se puede apreciar los trabajos combinan el enfoque heurístico con las reglas de despacho (*dispatching rules*).

Como se mencionó anteriormente, este trabajo está inspirado en un problema real de una empresa del sector automotriz. Por lo cual, enseguida se describen varios trabajos que han sido inspirados en casos reales en diferentes áreas del sector industrial.

2.3.2 El problema del HFS en la industria

El problema del Hybrid Flow Shop ha sido identificado en muchas áreas, en esta sección se presentará una variedad de trabajos aplicados a la industria. Estos trabajos tienen en común que sus procesos cuentan con múltiples máquinas en cada etapa (por ejemplo, Brah y Hunsuckerl [9]). En esta tesis se hará énfasis en los trabajos que se desarrollan en ambientes de manufactura flexible (Zijm y Nelissen [53]), debido a que describe un ambiente similar. Después de esta sección, se presentarán los trabajos relacionados con líneas de producción con etapas de ensamble.

Un trabajo interesante en la industria manufacturera de filmes fotográficos es el presentado por Tsubone et al. [51]. En este trabajo se analiza la programación de producción la cual presenta el ambiente de un HFS formado por una etapa de corte de rollos de películas y posteriormente la fase de entrelazado, así hasta finalizar con la etapa de envoltura de hojas del filme. El problema es subdividido en dos problemas y resuelto mediante un algoritmo heurístico.

La industria de la construcción es una de las áreas donde frecuentemente se ven aplicaciones de HFS. La producción de órdenes de blocks de concreto se analizan como un sistema de múltiples etapas en serie. El problema analizado en Grabowski y Pempera [23] considera ocho etapas, y en la mayoría de estas se cuentan con una sola máquina. Ellos proponen un algoritmo basado en Tabu Search para la solución de este problema.

El trabajo propuesto por Lin y Liao [39] analiza una línea de producción de una empresa que se dedica a elaborar etiquetas adhesivas. El objetivo de ellos consiste en minimizar la tardanza máxima ponderada. En la primer etapa se cuenta con una máquina para pegar el material de la superficie y crear las etiquetas, mientras la segunda consiste en máquinas paralelas encargadas del corte de las etiquetas. Ellos utilizan un algoritmo heurístico, en este algoritmo determinan la secuencia mediante el algoritmo propuesto por Gupta y Tunc [27], posteriormente mejoran el

valor objetivo de la solución mediante una búsqueda Tabú.

En el sector financiero existen problemas relacionados con el HFS. En el trabajo de Bertel y Billaut [8] resuelven un problema relacionado con el cobro de cheques de una compañía. Se consideran tres etapas y operaciones como la Captura de ID., empaquetamiento, escaneo, impresión y captura final de datos. Es interesante ver el impacto de su función objetivo, ya que ellos consideran minimizar el número promedio de trabajos tardíos, esta función está asociada con disminuir la inmovilización de dinero. Este trabajo cuenta con el supuesto de recirculación de trabajos. Para resolver este problema utilizan un algoritmo voraz y un genético.

El trabajo propuesto de Ruiz y Vázquez-Rodríguez[45] en el cual adapta un algoritmo genético para resolver un HFS, ha sido implementado en sistemas de producción de compañías dedicadas a los azulejos de cerámica. Una característica importante en este problema es elegibilidad de algunos trabajos con máquinas, a este proceso en este trabajo se le conocerá como incompatibilidad. Cabe destacar que este mismo algoritmo ha sido encajado en software de sistemas de producción dedicados a la industria textil o de pinturas sintéticas.

Otro sector de aplicaciones de HFS, es la arquitectura de las redes en los centros de cómputo, especialmente donde se cuentan con servidores en paralelo. Un trabajo en esta área es el propuesto por Allahverdi y Al-Anzi [4], ellos desarrollan un algoritmo basado en la metaheurística de Recocido Simulado para un problema de HFS con un objetivo basado en el tiempo promedio de respuesta del sistema de computo.

Otras dos implementaciones de la metaheurística Tabu Search en HFS pueden verse en los trabajos de Chen et al. [11] y Janiak et al [34]. El primer trabajo está relacionado con la programación de equipos de trabajo de contenedores de un puerto. En este trabajo se consideran los buffers con capacidades finitas, tiempos de preparación en las máquinas. El segundo trabajo es motivado en el estudio de una fábrica de fundición de hierro en la cual existe cumplimiento de pedidos, en base a

esto consideran una función objetivo relacionando costos de almacenamiento y de incumplimiento. Ellos proponen tres algoritmos basados en Tabu Search, Recocido Simulado y un hibrido de los dos anteriores, posteriormente comparan sus resultados mediante dos métricas, una de porcentaje de desviación de las soluciones y la otra tiempo promedio de ejecución.

En el trabajo propuesto por Voss y Witt [52], es interesante ver que la línea de producción de la compañía de acero que analizan cuenta con dieciséis etapas, pues es similar a la línea de producción estudiada en esta tesis. Ellos resuelven su problema mediante un algoritmo heurístico basado en reglas de secuenciación. El objetivo se basa en minimizar la tardanza ponderada.

Se han encontrado pocos trabajos relacionados con el HFS y la capacidad en los buffers, uno de ellos es el propuesto por Tavakkoli-Moghaddam et al. [49], en este trabajo proponen un algoritmo memético para la solución del HFS.

En el trabajo propuesto por Gicquel et al. [19], se estudia un bio-proceso industrial, el cual cuenta con etapas de fermentación, mezcla de nutrientes y filtración de nutrientes. Ellos proponen un modelo matemático y un algoritmo basado en ramificación y acotamiento para la solución del problema.

Después de este breve recorrido por los diferentes sectores industriales donde han surgido problemas de HFS, se pudo observar que las técnicas de solución más usadas son Tabu Search, Recocido Simulado y Algoritmos Geneticos. Enseguida se realiza una descripción de los trabajos relacionados con el HFS y el ensamble.

2.3.3 HFS en líneas de ensamble

El problema abordado en esta tesis es encontrado en una línea de producción de ensamble. Algunos trabajos previos de líneas de ensamble sugieren programar las sub-líneas de producción por separado. En este trabajo se decidió abordar la

problemática como una sola programación. A continuación mostramos los trabajos de HFS con ensamble más sobresalientes de la literatura que tienen algunas semejanzas con el problema central de la tesis.

Uno de los primeros trabajos propuestos fue el de Agnetis et al. [2], en este trabajo estudian una línea de producción de ensamble de automóviles. Este sistema tiene dos procesos: mecánica y tableros. El primer proceso cuenta con seis etapas y con buffer en cada etapa. Mientras que el segundo proceso cuenta con dos subáreas y un total de cinco etapas. Ellos establecen como función objetivo minimizar el número de componentes tardíos y solucionan el problema mediante *dispatching rules* y técnicas de simulación.

Otro trabajo interesante es el propuesto por Leey Vairaktarakis [37], ellos comparan tres diseños de una línea de producción de ensamble. El tercer diseño es el que se relaciona con nuestro trabajo, este diseño es considerado como un Flexible flow Shop. En estos diseños se cuenta únicamente con dos etapas, con m máquinas en la primer etapa y una sola en la etapa de ensamble. Ellos analizan el problema desde el enfoque de programación dinámica y consideran como función objetivo el makespan, además que proponen diferentes heurísticos para solucionar el problema.

Diversos trabajos consideran únicamente dos etapas en las líneas de producción. En He et al. [29] estudian una línea con etapas de mecanizado y ensamble. Contemplan características similares a nuestro trabajo, como: tiempos de preparación, sin capacidad en los buffers, tareas sin interrupción y como objetivo el makespan, sin embargo, ellos no contemplan la incompatibilidad y establecen un conjunto de restricciones diferentes a nuestro modelo para la secuenciación de los tiempos entre las dos etapas. Ellos proponen algunos algoritmos heurísticos para la solución del problema. En este caso es mucho más sencillo establecer la llegada simultánea en dos etapas de dos sub-líneas de producción, por lo cual, la llegada simultánea no es considerada en esta etapa.

Otro trabajo que involucra dos etapas y múltiples máquinas en una ambiente

de ensamble es el propuesto por Sung et al. [48]. Ellos proponen algoritmos basados en *dispatching rules* y heurísticas basados en ramificación y acotamiento.

En el trabajo de Sawik et al. [47] analizan la programación de una línea de producción de placas de circuitos impresos (PCB). Estos problemas son muy frecuentes en empresas dedicadas a la tecnología. En este trabajo el HFS cuenta con capacidad en los buffers y cada uno de estos es considerado como etapa. En total se consideran alrededor de trece etapas. Ellos proponen un modelo matemático, el cual consideran varios escenarios que pueden presentarse, como: la capacidad limitada en los buffers y la recirculación de trabajos, entre otros. Su modelo busca minimizar el tiempo total de flujo.

Un trabajo que involucra más decisiones dentro de la cadena de suministro es el propuesto por Torabi et al. [50]. En este trabajo se decide la cantidad de elementos a producir y la programación de producción. Sus objetivos son minimizar el costo promedio de almacenamiento, de preparación en máquinas y de transportación. El modelo matemático de este problema es no lineal. Desarrollan un algoritmo genético híbrido con búsqueda local y un método enumerativo para resolver el problema. En la decisión de la secuenciación de tareas, se observa que su modelo está basado en la posición de la tarea en la máquina, esto difiere al modelo propuesto en esta tesis.

De acuerdo con la revisión realizada en esta tesis, los trabajos con más características similares al nuestro pueden ser vistos en [15] - [16]. En el trabajo propuesto por Fattahi et al. [15] estudian un HFS con una etapa de ensamble. En este trabajo no se contemplan los tiempos de preparación en las máquinas, la incompatibilidad de productos y además los autores cosideran únicamente tres etapas, contando la etapa de ensamble y proponen un algoritmo basado en Johnson para resolver el problema. En el segundo trabajo Fattahi et al. [14] estudia el mismo problema presentado en Fattahi et al. [15], teniendo como variante la integración de tiempos de preparación (setup times) y operaciones de cambio (setup operation). Además, introducen un nuevo concepto de integración de operaciones y tiempos de cambio en una misma

máquina de un producto, a este concepto le llaman bloques. Ellos resuelven mediante algoritmos genéticos, recocido simulado y heurísticos basados en Jonhson. Por último, en Fattahi et al. [16] desarrollan un algoritmo basado en ramificación y acotamiento para el problema presentado en [15].

De acuerdo a la revisión de literatura presentada en este capítulo, se observa que nadie ha estudiado el problema presentado en esta tesis. Además, no se han propuesto adaptaciones de metaheurísticas como GRASP, BRKGA y Matheurístics para solucionar un HFS con ensamble. Las adaptaciones de estas metodologías son presentadas en el capítulo cuatro.

Capítulo 3

Planteamiento y formulación del problema

En este capítulo se describe el problema de programación de producción de una línea de ensamble estudiado en esta disertación, se mencionan los supuestos que fueron considerados en este problema y se presenta la formulación matemática. Por último, se concluye y se justifica el uso de metodologías heurísticas.

3.1 Planteamiento del problema

El problema consiste en la programación de producción de dos componentes que son ensamblados en una etapa de una línea de producción. Cada componente es producida en una sub-línea de producción. Cada sub-línea tiene una diferente cantidad de etapas y en cada una de estas se tiene un conjunto de máquinas en paralelo. La demanda de cada producto es divida en un conjunto de lotes (los cuales en su mayoría cuentan con el mismo tamaño). Cada lote debe ser procesado en una sola máquina de cada etapa, y deben pasar por todas y cada una de las diferentes etapas de cada sub-línea de producción. Nuestro objetivo consiste en minimizar el *makespan*, es decir, el tiempo total de producción. A continuación enlistamos los supuestos del problema:

- Se conocen los tiempos de procesamiento de los trabajos en cada máquina.
- Se cuentan con tiempos de preparación (setup times) en lotes de diferentes productos.
- Se tiene incompatibilidad de algunos productos con algunas máquinas.
- Se cuentan con buffers entre cada etapa, sin embargo, se tiene una amplia capacidad por lo cual no se considera.
- Cada máquina procesa un lote sin interrupción, es decir, una vez que el procesamiento de una tarea ha iniciado, este no puede ser interrumpido.
- Se sabe que la sub-línea de producción uno es hasta tres veces más rápida que la sub-línea de producción dos.
- El lote i de la componente uno y dos (correspondientes en cada sub-línea) deben llegar al mismo tiempo en la etapa de ensamble.

Gráficamente, en la figura 3.1 se muestra una representación de los productos, componentes y etapas. En la primer columna, se pueden observar tres productos, en la columna dos, se muestra la descomposición de cada producto en dos componentes. Después, cada componente se traslada a cada sub-línea de producción. La sub-línea de producción de la componente uno consta de dos etapas, con tres y dos máquinas respectivamente. La sub-línea de producción de la componente dos consta de tres etapas, con tres, una y dos máquinas respectivamente. La etapa ensamble es considerada como la etapa dos y tres de cada sub-línea de producción.

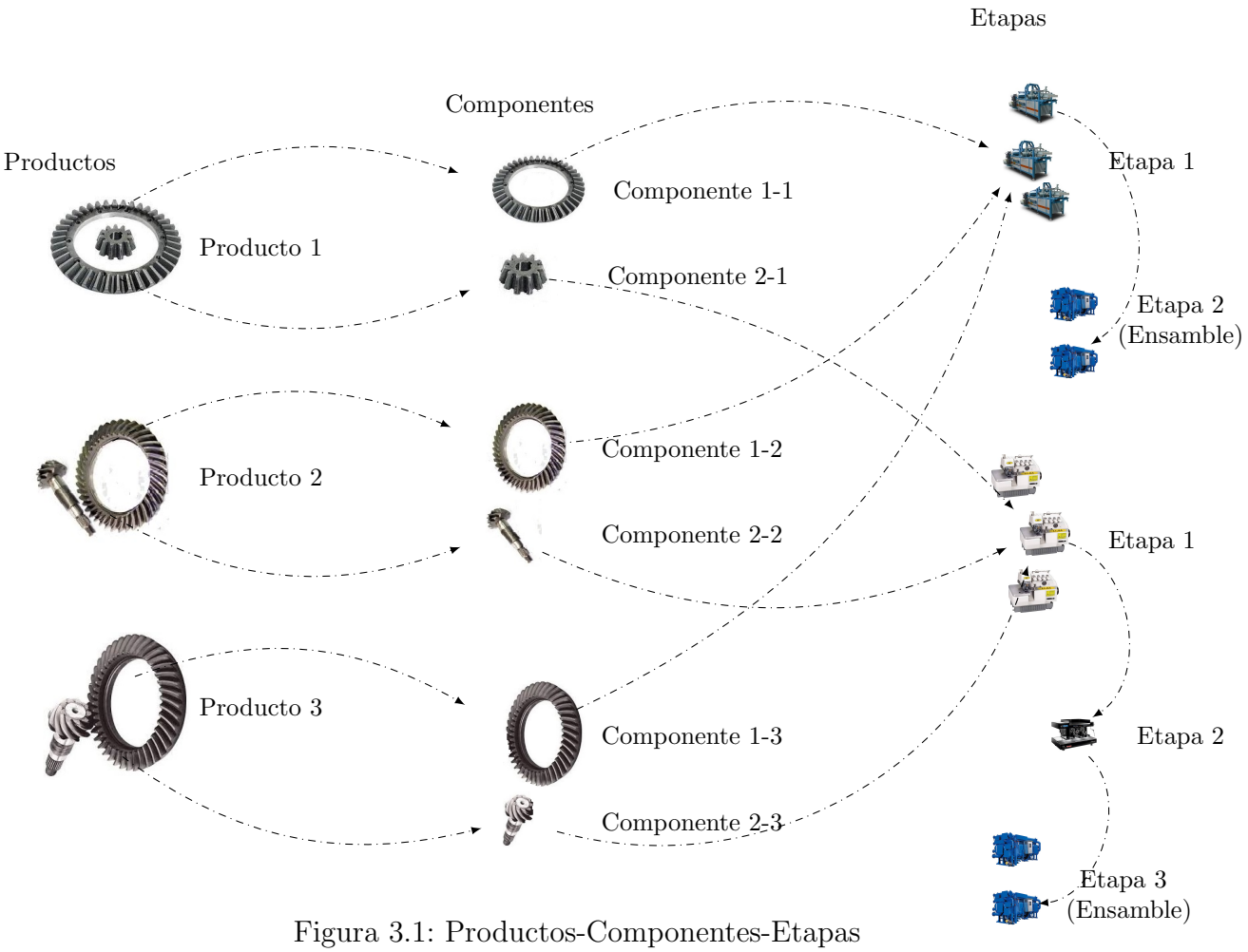


Figura 3.1: Productos-Componentes-Etapas

La línea de producción real, cuenta con quince y doce etapas en cada sublínea de producción, de una a tres máquinas en cada etapa etapa, más de cincuenta productos y dos componentes en cada producto. En este capítulo se realiza un modelo matemático que generaliza el problema de HFS con ensamble de k etapas, m máquinas, n productos y l componentes en una línea de ensamble.

3.2 Formulación matemática

En esta sección se presentan los elementos del modelo matemático, en primer orden los conjuntos e índices, después los parámetros y variables usadas en el modelo. Posteriormente se muestra el modelo y su descripción detallada.

3.2.1 Conjuntos, parámetros y variables

Conjuntos

N : Conjunto de productos, indexados por n.

I_n : Conjunto de lotes de cada producto n, indexado por i, j. (tareas)

0: Lote ficticio.

L : Conjunto de componentes, indexado por $l = \{l_1, l_2\}$.

K : Conjunto de etapas, indexado por k.

K_l: Subconjunto de etapas k del componente l.

a: Etapa de ensamble a.

M : Conjunto de máquinas, indexado por la máquina m.

$M_{k,l}$: Subconjunto de máquinas m de la etapa k, del componente l.

M_l: Subconjunto de máquinas m del componente l. $M_l = \{M_{1,l} \cup M_{2,l} \cup ... \cup M_{k,l}\}$.

Parámetros

$$e_{i,m}^{l} = \begin{cases} 1, & \text{Si es compatible el lote } i \text{ en la máquina } m \text{ del componente } l. \\ 0, & \text{De otro modo.} \end{cases}$$

$p_{j,m}$ Tiempo de procesamiento del lote j en la máquina m.

$\tau_{(i,j)}^{m,l}$ Tiempo de preparación del lote i al lote j en la máquina m del componente l.

Variables

$$y_{i,m}^{l} = \begin{cases} 1, & \text{Si se asigna el lote } i \text{ en la máquina } m \text{ del componente } l. \\ 0, & \text{De otro modo.} \end{cases}$$

$$x_{i,j}^{m,l} = \begin{cases} 1, & \text{Si el lote } i \text{ es secuenciado inmediatamente antes del lote } j \\ & \text{en la máquina } m \text{ del componente } l. \\ 0, & \text{De otro modo.} \end{cases}$$

$C_{i,k}^{l}$ Tiempo de completamiento del lote i en la etapa k del componente l.

$S_{i,j}^{k,l}$ Tiempo de iniciación del lote j secuenciado inmediatamente después del lote i en la etapa k del componente l.

C_{max} Variable auxiliar para el *makespan*.

3.2.2 Modelo matemático

Al modelo matemático presentado en esta sección se citará con las siglas *MM-HFS-E*. A continuación se presenta el modelo *MM-HFS-E* y la descripción de cada una de sus expresiones.

$$min \;\; C_{max} \tag{3.1}$$

s.t.:

$$C^l_{i,a} \leq C_{max} \qquad i \in I,\ l \in L,\ etapa\ final\ a \tag{3.2}$$

$$y^l_{i,m} \leq e^l_{i,m} \qquad i \in I,\ \ m \in M_l,\ \ l \in L \tag{3.3}$$

$$\sum_{m \in M_{k,l}} e^l_{i,m} * y^l_{i,m} = 1 \qquad i \in I,\ k \in K, l \in L \tag{3.4}$$

$$x^{m,l}_{i,j} - x^{m,l}_{j,i} \leq 1 \qquad i,j \in I,\ \ i < j,\ \ m \in M_l,\ \ l \in L \tag{3.5}$$

$$x^{m,l}_{i,j} \leq y^l_{j,m} \qquad i,j \in I,\ \ i < j,\ \ m \in M_l,\ \ l \in L \tag{3.6}$$

$$\sum_{j \in I} x^{m,l}_{0,j} = 1 \qquad m \in M_{k,l}, k \in K,\ l \in L \tag{3.7}$$

$$\sum_{i \in I \cup 0,\, k \neq i} x^{m,l}_{i,h} - \sum_{j \in I \cup 0,\, k \neq j} x^{m,l}_{h,j} = 0 \qquad h \in I,\ m \in M_{k,l}, k \in K,\ l \in L \tag{3.8}$$

$$\sum_{j \in I} x^{m,l}_{j,0} = 1 \qquad m \in M_{k,l}, k \in K,\ l \in L \tag{3.9}$$

$$\sum_{m \in M_{k,l}} \sum_{i \in I \cup 0} x^{m,l}_{i,j} = 1 \qquad j \in I,\ k \in K,\ l \in L \tag{3.10}$$

$$C^l_{j,k} \geq S^{k,l}_{i,j} + p_{j,m} * y^l_{j,m} + \tau^{m,l}_{(i,j)} * x^{m,l}_{i,j} - B * (1 - x^{m,l}_{i,j})$$

$$i,j \in I,\ i \neq j,\ m \in M_{k,l}, k \in K,\ l \in L \tag{3.11}$$

$$S^{k,l}_{i,j} \geq C^l_{j,k-1} \qquad i,j \in I,\ i \neq j,\ k \in K_l,\ k > 1,\ l \in L \tag{3.12}$$

$$S^{k,l}_{i,j} \geq C^l_{i,k} \qquad i,j \in I,\ i \neq j,\ k \in K_l,\ l \in L \tag{3.13}$$

$$C^{l_1}_{i,a} - C^{l_2}_{i,a} = 0 \qquad i \in I \tag{3.14}$$

$$x_{i,j}^{m,l_1} - x_{i,j}^{m,l_2} = 0 \qquad i,j \in I,\ i \neq j,\ m \in P_n \tag{3.15}$$

$$y_{i,m}^{l_1} - y_{i,m}^{l_2} = 0 \qquad i \in I,\ m \in M_{n_1,l} \cup M_{n_2,l},\ l \in L \tag{3.16}$$

$$x_{i,j}^{m,l}, y_{i,m}^{l} \in \{0,1\} \qquad i,j \in I,\ m \in M_{k,l}, k \in K,\ l \in L \tag{3.17}$$

$$C_{i,k}^{l}, S_{i,j}^{k,l}, C_{max} >= 0 \qquad i,j \in I,\ m \in M_{k,l}, k \in K,\ l \in L \tag{3.18}$$

La expresión (3.1) representa la función objetivo del Makespan, para encontrar C_{max} nos apoyamos en la desigualdad (3.2) la cual encuentra el tiempo final de todos los lotes en la última etapa. La desigualdad (3.3) implica la incompatibilidad de cada lote en cada máquina. El parámetro del lado derecho representa una matriz binaria. La igualdad (3.4) indica que cada lote debe ser asignado a una máquina en cada etapa. La expresión (3.5) indica que si el lote i es secuenciado antes del lote j en una máquina, no deberá permitirse realizar la secuencia del lote j y después el i. La desigualdad (3.6) establece que si el lote i no es asignado en la máquina m, las variables de secuenciación no deben ser activadas. Las ecuaciones (3.7)-(3.9) establecen la conservación de flujo. En las ecuaciones (3.7) y (3.9) respectivamente, implican el inicio y fin en cada máquina en un lote ficticio. Mientras que en la (3.8) establece la conservación de flujo. En la ecuación (3.10) forzamos a que cada lote debe ser secuenciado en una máquina de cada etapa.

La restricción (3.11) proporciona un orden de procesamiento correcto, evitando ciclos dentro de una misma máquina. El tiempo de completamiento de cada lote en cada etapa, es mayor o igual al tiempo inicial más su tiempo de producción y su setup time menos la penalización si no es activada la variable. El setup time depende de la combinación de lotes secuenciados, estas combinaciones son representadas por una matriz τ. El tiempo de iniciación de cada lote j en cada etapa depende de quién es mayor entre el tiempo del lote (terminación) secuenciado anteriormente i en esa misma máquina (3.13) o el tiempo de terminación del mismo lote j en una etapa anterior (3.12). Finalmente, la restricción (3.14) indica la igualación de tiempos en la etapa de ensamble, las restricciones (3.15)-(3.16) implican la igualación de variables de asignación y secuenciación en las de etapas posteriores a esta.

En el trabajo de Gupta [26], demuestran que un HFS con dos etapas de procesamiento, una y dos máquinas respectivamente en cada etapa, n trabajos y con función objetivo makespan, es NP-duro. Si el problema que se aborda en este trabajo se considera una sola componente en cada producto, dos etapas, una y dos máquinas y no cuenta con la etapa de ensamble, se tendría una versión similar al HFS mencionado en Gupta [26], y por lo cual se mantiene la complejidad ya demostrada.

El modelo *MM-HFS-E* es considerado como un modelo de programación lineal mixto entero (por sus siglas en ingles MILP) y fue implementado en CPLEX 12.5, se validó con instancias de diferentes tamaños, las cuales son descritas en la el capítulo cinco. Previamente se define el GAP se define como:

$$GAP = \frac{Valor\ encontrado - Mejor\ Valor}{Mejor\ Valor} \tag{3.19}$$

A continuación se presenta la Tabla 3.1, la cual indica la instancia, el mejor valor de la función objetivo y el GAP alcanzado por CPLEX en un tiempo de una hora.

Tabla 3.1: Resultados CPLEX

Instancia	CPLEX	GAP
1	1365.82	0.68
2	891.636	0.24
3	1297.93	0.73
4	1106.77	0.68
5	1194	0.70
6	-	-
7	3411.88	0.97
8	4722.18	1
9	4180.94	1
10	-	-
11	-	-
12	-	-
13	-	-
14	-	-
15	-	-
16	-	-
17	-	-
18	-	-
19	-	-
20	-	-

En la Tabla 3.1, se observa que valor del GAP proporcionado por CPLEX de la solución incumbente no bajaba de veinticinco por ciento en las instancias pequeñas (1-5). Para las instancias grandes (11-20) no se encontró solución, por lo cual, decidimos emplear metodologías heurísticas para encontrar buenas soluciones al problema.

3.3 Resumen del capítulo

En este capítulo se presenta la descripción del problema y el modelo matemático de programación lineal mixto entero. Se formula el problema bajo el enfoque de optimización entera mixta, donde las variables de decisión asignan los lotes a las máquinas y realizan la secuenciáción de estos lotes en cada una de las máquinas. Mientras que las variables auxiliares dan a conocer los tiempos de inicio y terminación de cada lote en cada etapa. Se toma en cuenta todos los supuestos planteados en esta sección, buscando minimizar el tiempo de terminación.

Capítulo 4

Metodologías de solución para el problema de HFS con ensamble

El modelo matemático presentado en el capítulo tres fue validado con instancias de diferentes tamaños y fue resuelto en el optimizador CPLEX 12.5. En las instancias más pequeñas se obtuvieron GAP's mayores de un veinticuatro por ciento, en un tiempo de ejecución de un día. Estas instancias cuentan con dos máquinas, dos etapas y alrededor de ocho trabajos en cada sub-línea de producción. En una instancia real, se tienen alrededor de catorce etapas en ambas sub-líneas de producción y más de cincuenta trabajos, por lo cual, se decidió implementar algoritmos heurísticos para la obtención de buenas soluciones en tiempo aceptable.

Este capítulo presenta diferentes metodologías de solución, la primera de ellas está basada en la metaheurística GRASP, la cual consiste en una fase constructiva y una búsqueda local. La segunda es una adaptación de la metaheurística BRKGA. Por último, se presenta una metodología basada en matheuristics, la cual aprovecha la estructura del modelo matemático y descompone el problema en asignación y secuenciación. Estas metodologías serán comparadas en un capítulo cinco.

4.1 Una metodología de solución basada en GRASP

La metodología GRASP fue desarrollada por Feo y Resende [17] en el año de 1989, al estudiar un problema de cobertura de alta complejidad. GRASP es el acrónimo de *greedy randomized adaptative search procedure* (procedimiento de búsqueda miope, aleatorizado y adaptativo). Este algoritmo consiste en dos fases, la fase de construcción y la fase de mejora.

Los principales características de GRASP para la fase de construcción son una función de evaluación miope, la selección al azar y un proceso de actualización adaptativo. La fase de construcción consiste en incorporar elementos en una solución parcial. Previamente se tienen un conjunto de elementos candidatos que pueden incorporarse a la solución parcial. Para cada elemento candidato se calcula el beneficio en la función miope $c(e)$. Posteriormente, se forma una lista de candidatos restringidos *(LRC)*, esta lista contiene elementos de alta calidad que cumplan la desigualdad (4.1). El parámetro c_* representa el valor más pequeño de la función miope al evaluar los candidatos. Mientras que c^* representa el valor más grande de la función miope. El valor de α toma valores del intervalo [0,1]

$$c(e) \leq c_* + \alpha(c^* - c_*). \tag{4.1}$$

De los elementos de la lista *(LRC)* se escogerá uno aleatoriamente y se ingresa a la solución parcial. Una vez agregado un elemento candidato a la solución se deberá recalcular los valores de la función miope, para realizar nuevamente el proceso de selección de la lista LRC, así el procedimiento adquiere la característica de adaptabilidad.

Al ejecutar el proceso anterior una cantidad de veces, se pretende que se genere

una cantidad diversa de soluciones. A cada una de estas soluciones se aplicará un proceso de búsqueda local. Como se menciona en Resende et al. [44], la búsqueda local consiste en explorar repetidamente una vecindad de solución en busca de encontrar una mejor solución. Primero se describe la nomenclatura del algoritmo, posteriormente el Algoritmo basado en GRASP (1) y en las siguientes secciones mostramos cada una de sus etapas.

Las figuras que se muestran en esta sección para ejemplificar los algoritmos tienen las siguientes características: los lotes de un mismo producto tienen un mismo color. Se tienen tres productos y un total de nueve lotes. Se cuenta con cuatro etapas y un total de ocho y nueve máquinas en cada sub-línea. En color rojo se muestra la etapa de ensamble, en la cual debe de existir una igualdad de lotes en cada posición en las dos sub-líneas de producción. La terna (a, b, c) esta formada por el parámetro a que indica la cantidad de elementos del lote, el parámetro b, el cual representa el producto al que pertenece el lote y el parámetro c indica el número del lote.

Para fines del algoritmo basado en GRASP, llamaremos inserción de un lote i al proceso de añadir un lote al final de la secuencia inicial de una máquina m. Véase las imágenes 4.1-4.2 en las que se inserta el lote de color verde.

Figura 4.1: Antes de la inserción

Figura 4.2: Después de la inserción

El *Algoritmo 1 basado en GRASP* consiste en realizar un total de iteraciones $Iter$. En cada iteración construiremos una solución parcial dos (x^2) y una solución parcial uno (x^1), correspondientes a la componente dos y uno respectivamente. La unión de estas dos soluciones parciales $x^1 \cup x^2$ forman la solución x. Después de la fase de construcción, la solución x entrará al método de *Búsqueda Local* para obtener la solución mejorada x_{new}. En cada iteración, el valor objetivo de la solución mejorada

$F_{mak}(x_{new})$ es comparado con el valor de la mejor solución $F_{mak}(X^*)$ encontrada hasta el momento. Al terminar todas las iteraciones, el algoritmo devuelve la mejor solución encontrada y su valor objetivo $(X^*, F_{mak}(X^*))$.

La siguiente nomenclatura representa los conjuntos e indices que estan interactuando en el algoritmo basado en GRASP, parte de ella es de la formulación matemática presentada en el capítulo tres.

Nomenclatura de conjuntos

N : Conjunto de productos, indexados por n.

L : Conjunto de componentes, indexados por l.

I : Conjunto de lotes, indexados por i.

I_n : Conjunto de lotes I de cada producto n, indexado por i_n.

$I_{n,a}$: Conjunto de lotes I de cada producto n en la etapa de ensamble a.

K_l : Conjunto de etapas del componente l, ordenadas desde el inicio al fin.

K_l^{-1} : Conjunto de etapas del componente l, ordenadas desde el fin al inicio.

M_i : Conjunto de máquinas del componente l, ordenadas desde el inicio al fin.

M_i^{-1} : Conjunto de máquinas del componente l, ordenadas desde el fin al inicio.

C : Conjunto de soluciones parciales.

F_n : Conjunto de funciones del subconjunto de lotes de cada producto.

F_s : Conjunto de funciones de makespan parcial.

En la siguiente sección se muestra una descripción de la nomenclatura que identifica a la solución x y sus diferentes movimientos.

Nomenclatura de soluciones:

x : Solución.

X^* : Mejor Solución.

x^l : Solución parcial del componente l.

$x^l_{m_i}$: Solución parcial del componente l, insertando el lote i en la máquina m

$x^l_{m_i^*}$: Solución parcial del componente l, insertando el lote i en la mejor opción de las máquinas m^*

$x^l_{m_{i,k}}$: Solución parcial del componente l, insertando el lote i en la máquina m de la etapa k

$x^l_{k_i}$: Solución parcial del componente l, insertando el lote i en todas las máquinas de la etapa k

$x^l_{i_n}$: Solución parcial del componente l, insertando todos los lotes i del producto n: I_n

x_{new} : Solución mejorada después de la Búsqueda Local

Por último, se muestran los parámetros que interactúan en el algoritmo y posteriormente su pseudocódigo principal.

Nomenclatura de parámetros:

$Iter$: Total de iteraciones.

a : Etapa de ensamble.

$\lambda^m_{i,j}$: Tiempo perdido del lote i al lote j secuenciado en la máquina m

α : El nivel de aleatoridad de la lista restringida de candidatos LRC.

Algoritmo 1 Basado en GRASP

$X^* \leftarrow \emptyset$

$F_{mak}(X^*) \leftarrow \infty$

para $i <= Iter$ **hacer**

$x^2 \leftarrow$ Construcción de la solución parcial dos ()

$x^1 \leftarrow$ Construcción de la solución parcial uno ()

$x \leftarrow x^1 \cup x^2$

$x_{new} \leftarrow Busqueda\ Local(x)$

si $F_{mak}(x_{new}) \leq F_{mak}(X^*)$ **entonces**

$X^* \leftarrow x_{new}$

fin si

fin para

$(X^*, F_{mak}(X^*))$

En las siguientes secciones se describen detalladamente cada uno de los procedimientos utilizados en el *Algoritmo 1 basado en GRASP*.

4.1.1 Fase constructiva

La fase constructiva es dividida en dos subprocesos. El primero se denomina Construcción de la solución parcial dos, este consiste en crear la programación de producción de la componente dos. El segundo se denomina Construcción de la solución parcial uno, y análogamente al primero, consiste en programar la producción de la componente uno.

Debido a que la sub-línea de producción de la componente uno es más rápida que la de la componente dos, se decidió construir primero la solución parcial dos y una vez terminado esta, construir la de la solución uno en sentido inverso. Después de construir la solución parcial dos se conocerán los tiempos de la etapa de ensamble, así, estos servirán como datos de entrada para la construcción de la solución parcial

uno, la cual se realizará desde la etapa ensamble hasta la etapa inicial.

Algoritmo 2 construcción de la solución parcial dos. En cada solución se realiza lo siguiente: Para cada lote i a probar, se calcula una función parcial de tiempo en cada máquina m. Esta función 4.2 consiste en minimizar el tiempo final $T_f(x_m^2)$ menos el tiempo inicial $T_0(x_m^2)$ más la suma de los tiempos perdidos $\lambda_{i,j}^m$:

$$F_m \leftarrow T_f(x_m^2) - T_0(x_m^2) + \sum \lambda_{i,j}^m. \tag{4.2}$$

Así hasta encontrar la mínima inserción paralela $x_{m^*}^2$ en cada etapa k.. En las siguientes figuras (4.3, 4.4, 4.5, 4.6) se muestra la evaluación de la función F_m en la etapa uno, insertando el lote con la terna $(25, 2, 5)$ (color verde) y se encuentra la mínima inserción $x_{m^*}^2$.

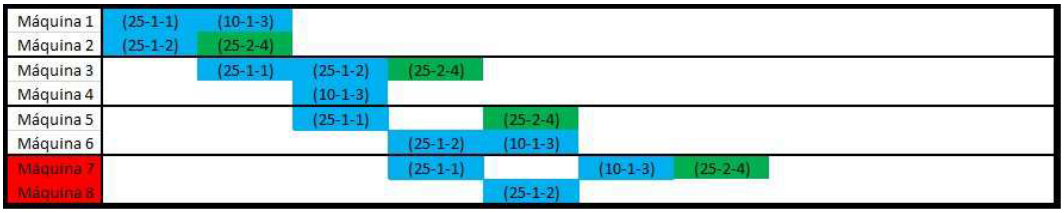


Figura 4.3: Imagen inicial

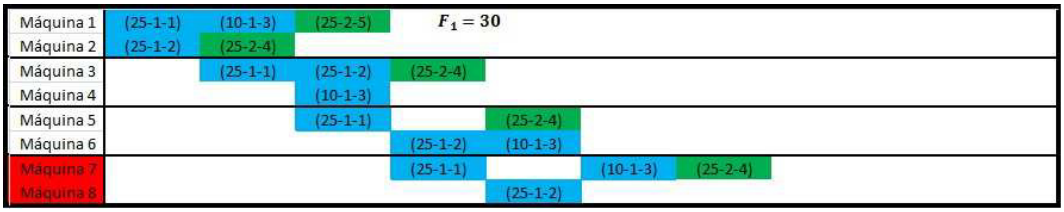


Figura 4.4: Inserción en máquina uno

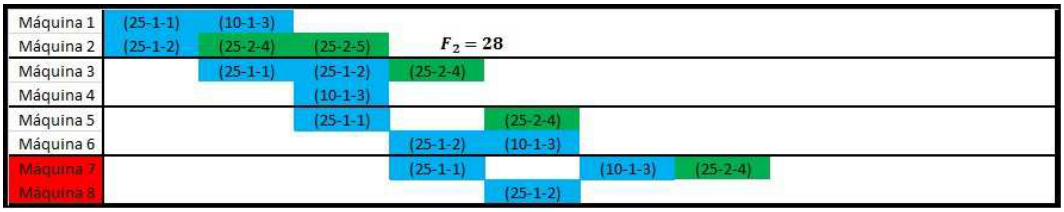


Figura 4.5: Inserción en máquina dos

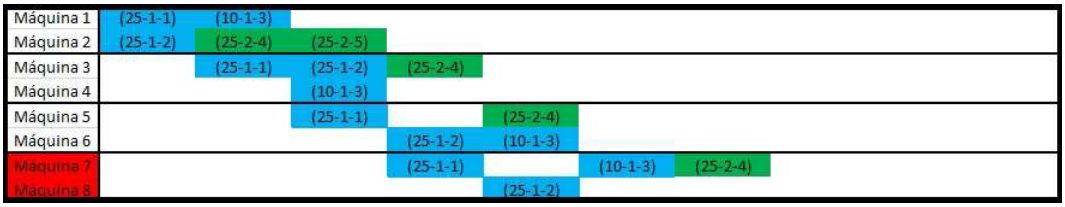


Figura 4.6: Mejor inserción: $x^2_{m^*}$

Lo anterior lo realizaremos para todos los lotes i de cada producto n compatibles con cada máquina m de cada etapa k. Como se muestra en la siguiente figura (4.7) en la cual el lote $(25, 2, 5)$ ya fué insertado en todas las etapas.

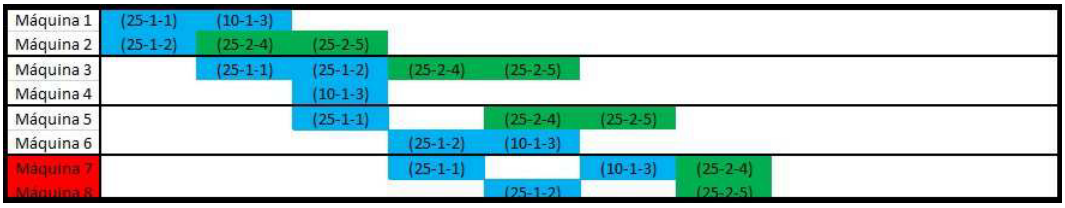


Figura 4.7: Lote $(25, 2, 5)$ insertado en todas las etapas

Después de que el conjunto de lotes candidatos de un mismo producto i_n es probado en todas las etapas, este será evaluado en una función llamada Makespan parcial $F_{mak}(x^2_{i_n})$, es decir, se calculará el tiempo final en la última etapa hasta la inserción del último elemento de este conjunto. Para este trabajo será la función miope. A continuación se muestran las figuras (4.8)-(4.9) que representan la evaluación de los lotes de cada producto en la función $F_{mak}(x^2_{i_n})$

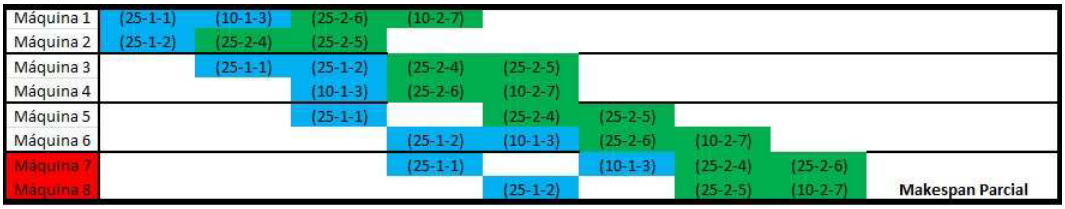


Figura 4.8: Lotes del producto dos (verde)

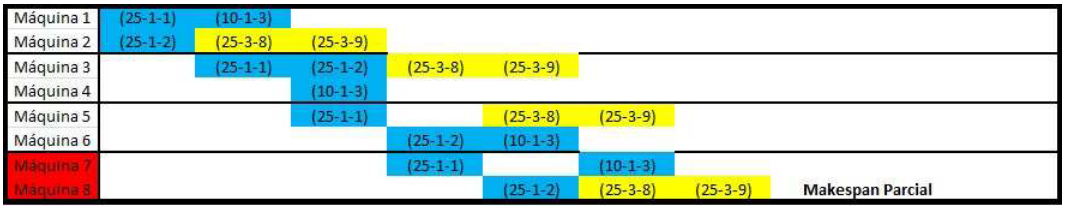


Figura 4.9: Lotes del producto tres (amarillo)

Después de evaluar todos los conjuntos de lotes de todos productos, se hará una lista restringida de candidatos LRC, y seleccionaremos un candidato al azar (un conjunto de lotes de un producto) para realizar una inserción en la solución dos. La lista queda definida de la siguiente manera:

$$LRC \leftarrow \{x_{i_n}^2 \in C : F(x_n^2) \leq F_{min} + \alpha(F^{max} - F_{min})\}. \tag{4.3}$$

Este último proceso se realizará hasta terminar de insertar todos los candidatos. Como ya lo mencionamos, para el *Algoritmo 3 construcción de la solución parcial dos* se necesita saber el tiempo de cada lote en la etapa de ensamble, por lo cual, igualaremos los tiempos de inicio y fin de la etapa de ensamble de la solución parcial dos con la solución parcial uno. Enseguida se presenta el pseudocódigo del *Algoritmo 2 construcción de la solución parcial dos*.

Algoritmo 2 Construcción de la solución parcial dos

mientras $I_n \neq \emptyset$ **hacer**

para $n \in N$ **hacer**

para $i \in I_n$ **hacer**

para $k \in K_2$ **hacer**

para $m \in M_k$ **hacer**

si *i es compatible con m* **entonces**

$x^2_{m_i} \leftarrow x^2_{m_i} \cup i$: Se prueba el lote i en cada máquina m

$F_m \leftarrow T_f(x^2_{m_i}) - T_0(x^2_{m_i}) + \sum \lambda^m_{i,j}$: Se calcula la función F_m en cada máquina

fin si

fin para

$F_k(x^2_{m^*_i}) \leftarrow \text{mín}\, F_m(x^2_{m_i})$: Se obtiene el valor más pequeño de la función F_m en cada etapa k

$x^2_{k_i} \leftarrow x^2_{k_i} \cup x^2_{m^*_i}$: En cada etapa k se actualiza la solución parcial x^2_k

fin para

$x^2_{i_n} \leftarrow x^2_{k_l}$: Se actualiza la solución $x^2_{i_n}$ para cada conjunto de lotes de cada producto i_n

fin para

$C \leftarrow C \cup x^2_{i_n}$: Al terminar la inserción en todas las etapas, se actualiza el conjunto C con la solución parcial $x^2_{i_n}$

$F_n \leftarrow F_n \cup F_{mak}(x^2_{i_n})$: Se calcula el *Makespan Parcial* $F_{mak}(x^2_{i_n})$ y se guarda en F_n

fin para

$F^{max} \leftarrow \text{máx}\,(F_n)$: Se calcula el valor máximo de F_n de todos los conjuntos de lotes de cada producto n

$F_{min} \leftarrow \text{mín}\,(F_n)$: Se calcula el valor mínimo de F_n de todos los conjuntos de lotes de cada producto n

$LRC \leftarrow \{x^2_{i_n} \in C : F(x^2_n) \leq F_{min} + \alpha(F^{max} - F_{min})\}$: Se calcula la LRC

Se selecciona de forma aleatoria $x^2_{i_n}$ un conjunto de lotes de un producto i_n de la lista LRC

$x^2 \leftarrow x^2 \cup x^2_{i_n}$: Se ingresa el conjunto de lotes de un producto i_n a la solución parcial x^2

$I_n \leftarrow I_n \setminus i_n$: Se elimina el conjunto de lotes del producto i_n insertado

fin mientras

$T_0(x^1_a) = T_0(x^2_a)$: Se actualiza los tiempos iniciales de la etapa de ensamble de la solución parcial uno

$T_f(x^1_a) = T_f(x^2_a)$: Se actualiza los tiempos finales de la etapa de ensamble de la solución parcial uno

Constructivo solución parcial uno [3]. En el algoritmo de construcción de la solución parcial uno, realizaremos un proceso similar a la construcción anterior, sin embargo, este proceso se hará en sentido inverso y únicamente para etapas anteriores a la etapa de ensamble. En cada etapa seleccionaremos el lote i con el menor tiempo al insertarse, restando el tiempo final $T_f(x_m^1)$ menos el tiempo inicial $T_0(x_m^1)$ en cada máquina.

$$F_m(x_m^1) \leftarrow T_f(x_m^1) - T_0(x_m^1). \tag{4.4}$$

En las siguientes figuras (4.10)- (4.11) se muestra la inserción del lote $(10, 2, 7)$ (color verde) en las etapas anteriores a la fase de ensamble en la sub-línea del componente uno.

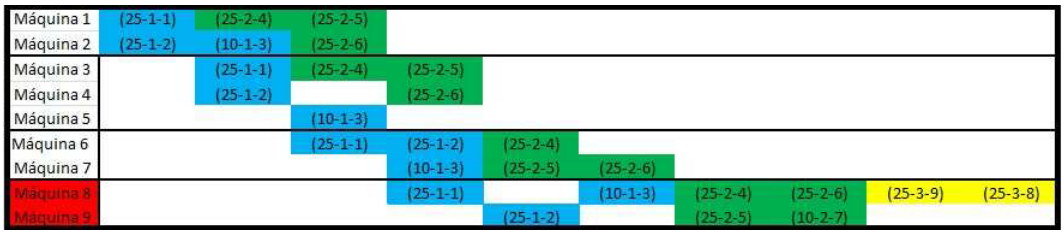


Figura 4.10: Antes de la inserción del lote $(10, 2, 7)$ en sub-línea del componente uno

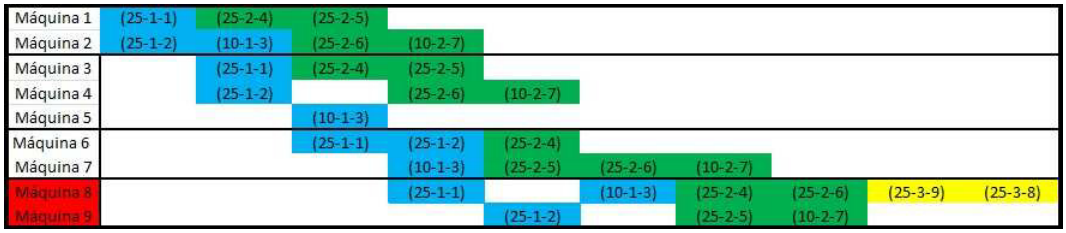


Figura 4.11: Después de la inserción del lote $(10, 2, 7)$ en sub-línea del componente uno

Es importante señalar que el orden en que los lotes son insertados en la solución parcial uno es igual al orden en que fueron insertados en la etapa de ensamble de la solución parcial dos.

En cada inserción se puede tener alguna infactibilidad de tiempo, esto sucede cuando el tiempo inicial de un lote $T_0(x_{m_{k+1}}^1)$ insertado en una máquina de la etapa

$k+1$, es menor que el tiempo final del lote antecesor $T_f(x^1_{m_k})$ insertado en una máquina de la etapa anterior k. Para corregir esta infactibilidad se debe restar la *diferencia* del tiempo final menos el tiempo inicial $(T_f(x^1_{m_k}) - T_0(x^1_{m_{k+1}}))$ a todos los tiempos (iniciales y finales) de las máquinas de las etapas anteriores a k. Esto se muestra en el pseudocódigo *Ajuste de Tiempos [4].*

Las figuras que a continuación se muestran (4.12)-(4.13), representan la construcción de una solución inicial.

1	(25-1-1)	(10-1-3)	(25-2-6)	(10-2-7)						
2	(25-1-2)	(25-2-4)	(25-2-5)	(25-3-8)	(25-3-9)					
3		(25-1-1)	(25-1-2)	(25-2-4)	(25-2-5)					
4			(10-1-3)	(25-2-6)	(10-2-7)	(25-3-8)	(25-3-9)			
5			(25-1-1)		(25-2-4)	(25-2-5)	(25-3-8)			
6				(25-1-2)	(10-1-3)	(25-2-6)	(10-2-7)	(25-3-9)		
7				(25-1-1)		(10-1-3)	(25-2-4)	(25-2-6)	(25-3-9)	(25-3-8)
8					(25-1-2)		(25-2-5)	(10-2-7)		

Figura 4.12: Construcción de solución parcial uno

1	(25-1-1)	(25-2-4)	(25-2-5)	(25-3-9)						
2	(25-1-2)	(10-1-3)	(25-2-6)	(10-2-7)	(25-3-8)					
3		(25-1-1)	(25-2-4)	(25-2-5)	(25-3-9)					
4		(25-1-2)		(25-2-6)	(10-2-7)					
5			(10-1-3)			(25-3-8)				
6			(25-1-1)	(25-1-2)	(25-2-4)	(25-3-9)				
7				(10-1-3)	(25-2-5)	(25-2-6)	(10-2-7)	(25-3-8)		
8				(25-1-1)		(10-1-3)	(25-2-4)	(25-2-6)	(25-3-9)	(25-3-8)
9					(25-1-2)		(25-2-5)	(10-2-7)		

Figura 4.13: Construcción de solución parcial dos

Algoritmo 3 Construcción de la solución parcial uno

para $i \in I_{a,n}$ **hacer**

para $k \in K_1^{-1}$ **hacer**

para $m \in M_k^{-1}$ **hacer**

si *i compatible con m* **entonces**

$x^1_{m_{i,k}} \leftarrow x^1_{m_k} \cup i$: Se prueba el lote i en cada máquina m

si $T_0(x^1_{m_{i,k+1}}) \geq T_f(x^1_{m_{i,k}})$ **entonces**

$F_m(x^1_{m_{i,k}}) \leftarrow T_f(x^1_{m_{i,k}}) - T_0(x^1_{m_{i,k}})$: Se calcula la función tiempo final menos tiempo inicial en una máquina

si no

$diferencia = T_f(x^1_{m_{i,k}}) - T_0(x^1_{m_{i,k+1}})$: Se calcula la diferencia que provoca la infactibilidad

$AjustedeTiempos(T(x^1_{m_{i,k}}), diferencia)$ Se ajusta los tiempos de las máquinas anteriores

$F_m(x^1_{m_{i,k}}) \leftarrow T_f(x^1_{m_{i,k}}) - T_0(x^1_{m_{i,k}})$

fin si

fin si

fin para

$x^1_{m^*_{i,k}} \leftarrow \text{mín}\, F_m(x^1_{m_{i,k}})$: Se encuentra la inserción más pequeña en cada etapa

$x^1_k \leftarrow x^1_k \cup x^*_{m_{i,k}}$: Se actualiza la solución en cada etapa

fin para

$x^1 \leftarrow x^1 \cup x^1_k$

fin para

Algoritmo 4 Ajuste de Tiempo

$\beta : m_1, m_2, ..., m_{k-1}$

para $m \in \beta$ **hacer**

para $i \in I_m$ **hacer**

$T_f(x_{m_i^l}) = T_f(x_{m_i^l}) - diferencia$

$T_0(x_{m_i^l}) = T_0(x_{m_i^l}) - diferencia$

fin para

fin para

4.1.2 Búsqueda local

Las soluciones obtenidas con los métodos de *Algoritmo 2 construcción de la solución parcial dos* y *Algoritmo 3 construcción de la solución parcial uno*, se mejoran con el método de búsqueda local. Como se mencionó anteriormente, este es un proceso iterativo, en cada iteración se realiza un movimiento buscando minimizar la función objetivo de la solución actual dentro de un vecindario.

Como se construyeron las soluciones, si hay lotes de un mismo producto dentro de una máquina, estos deben de estar secuenciados continuamente. Por ejemplo, si un conjunto de lotes de un producto tiene tres elementos, estos deberán tener las posiciones $i, i+1, i+2$ dentro de la secuencia. Debido a lo anterior, se decidió aprovechar la estructura de la solución y realizar un movimiento inter-máquina, es decir, dentro de la misma máquina. El movimiento consiste en lo siguiente:

Supongamos que se tiene la siguiente secuencia dentro de una máquina:

$$\{\ l_{n_1}, l_{n_2}, \ldots, l_{n_{d-1}}, l_{n_d}\ \}.$$

Cada elemento l_{n_d} representa un conjunto de lotes de un producto n. Se remueve el conjunto de lotes de la primera posición l_{n_1} y se inserta en la última posición de la máquina, en este caso la última posición pertenece a l_{n_d}. Todos los demás conjuntos de lotes se recorrerán una posición. Después de este movimiento tendremos la siguiente nueva secuencia:

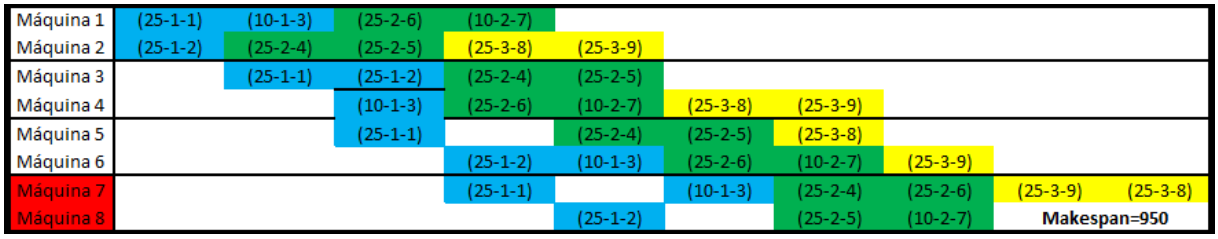
$$\{\ l_{n_2}, l_{n_3}, \ldots, l_{n_{d-1}}, l_{n_d}, l_{n_1}\ \}.$$

Este movimiento se realizará en todas las máquinas de todas las etapas las cuales tengan el mismo conjunto de lotes l_{n_1}. Una iteración se considera después de cambiar todos los lotes de un mismo producto en todas las etapas y actualizar los tiempos de las secuencias. Para actualizar los tiempos de las secuencias, primero se actualizará los tiempos de la solución parcial dos, desde la etapa de inicio hasta la etapa ensamble

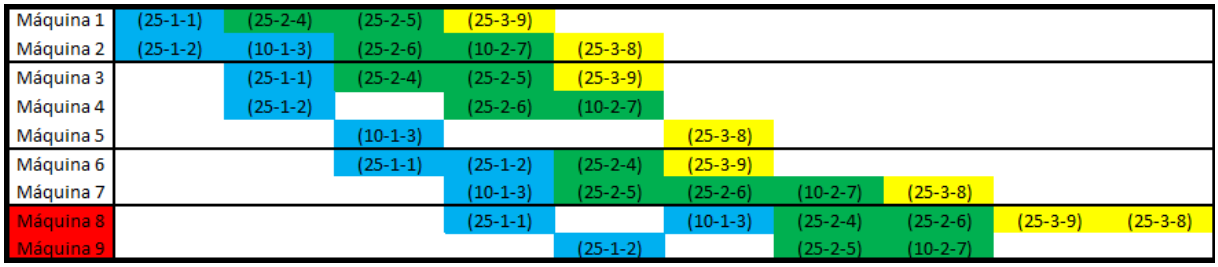
y posteriormente los tiempos de la solución parcial uno, desde una etapa anterior a la de ensamble hasta la etapa de inicio. Lo anterior es análogo a lo realizado en el método de construcción.

Después de realizar este movimiento con el conjunto de lotes l_{n_1} en todas las etapas, esto se repetirá de forma iterativa con todos los demás conjuntos de lotes hasta terminar en la misma posición inicial.

En las siguientes figuras (4.14a)(4.15a) (4.16a) se muestra la búsqueda local aplicada a la solución inicial (4.14a). En este ejemplo la mejor solución de acuerdo al valor del Makespan es la figura (4.15a).

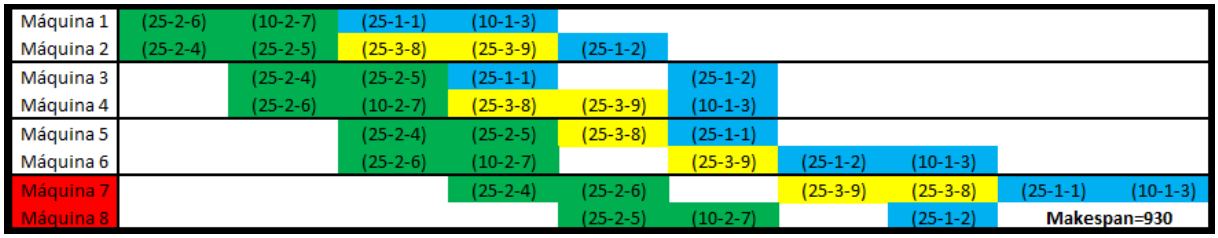


(a) Sub-línea componente dos

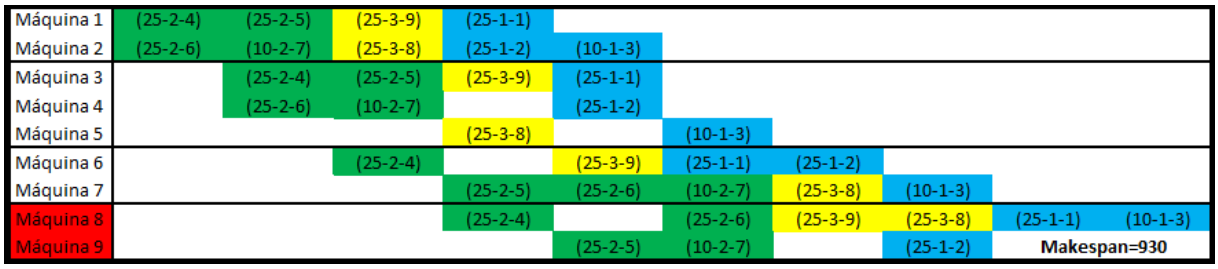


(b) Sub-línea componente uno

Figura 4.14: Solución inicial

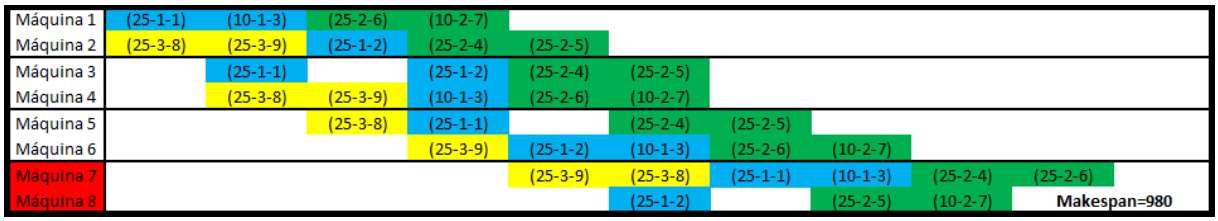


(a) Sub-línea componente dos

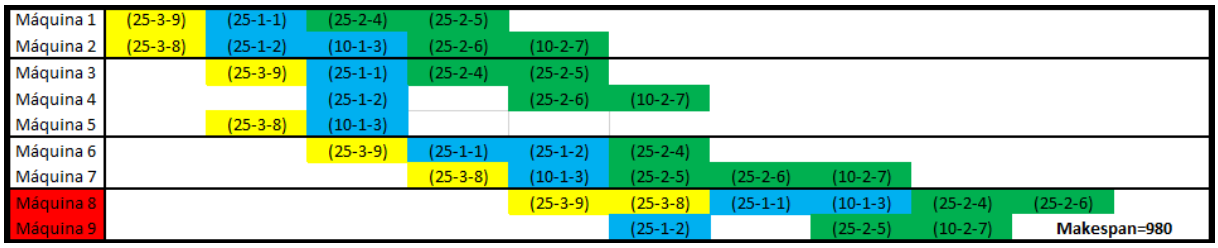


(b) Sub-línea componente uno

Figura 4.15: Iteración uno búsqueda local



(a) Sub-línea componente dos



(b) Sub-línea componente uno

Figura 4.16: Iteración dos búsqueda local

4.2 Una adaptación de la metaheurística BRKGA

Los Algoritmos Genéticos (por sus siglas en inglés GA) son algoritmos basados en procesos genéticos de organismos biológicos. Se basan en la supervivencia de los individuos más fuertes y el intercambio de información entre ellos. Estos algoritmos fueron introducidos por Holland [31] en el año de 1975. Estos algoritmos consisten en realizar un conjunto de generaciones, en las cuales se crea un conjunto de individuos (cromosomas) llamados población. Cada individuo (solución) debe ser codificado en una cadena de genes, estos pueden ser números enteros, los cuales son modificados a través de operaciones básicas como reproducción, cruzamiento y mutación.

La reproducción consiste en conservar los mejores individuos de la actual generación, esta selección se realiza a través de la evaluación de los individuos en una función llamada aptitud. El cruzamiento consiste en elegir dos individuos de forma aleatoria y combinar sus genes para producir un nuevo individuo. La mutación consiste en la modificación de algunos genes de ciertos individuos.

Una llave aleatoria es la representación de una solución en una cadena de valores aleatorios dentro del intervalo $(0, 1]$. En el trabajo de Bean [5], se proponen las llaves aleatorias para la codificación de soluciones de algoritmos basados en GA. A cada elemento de una llave aleatoria es llamado alelo. El algoritmo RKGA inicia generando una población P de valores aleatorios entre cero y uno. Después, los individuos de esta población pasan por un conjunto de operadores como reproducción, cruce y mutantes, los cuales tienen diferencias a los operadores clásicos. Algunas diferencias de estos operadores son las siguientes.

El operador de reproducción clasifica las soluciones en dos conjuntos: el conjunto elite P_e y el conjunto no elite P'_e. Estos conjuntos son clasificados en base a su aptitud. El conjunto elite se actualiza en cada generación y permanece en la

siguiente. El operador de cruce selecciona de forma *aleatoria* dos individuos de la población y busca generar un nuevo individuo. En lugar de realizar el operador de mutación (como en los GA), el cual modifica algunos genes de ciertos individuos, se busca generar nuevos individuos de la misma forma que fué generada la población inicial. Más adelante se detallarán estos operadores.

Una evolución del RKGA son los algoritmos genéticos de llaves aleatorias sesgadas (por sus siglas en inglés BRKGA), los cuales fueron introducidos por Goncalves y Resende [22] en el año 2009. Estos algoritmos difieren al RKGA en la manera en que seleccionan los dos padres en el operador de cruzamiento. El algoritmo BRKGA selecciona una llave aleatoria del conjunto elite y otra del no elite, de esta manera, al efectuar el operador de cruce con probabilidad sesgada hacia el conjunto elite, se busca conservar mayor cantidad de individuos de este conjunto.

Esta metaheurística no ha sido implementada en problemas de HFS, por lo cual se decidió implementar esta metodología al problema presentado en este trabajo. A continuación se describe este algoritmo y posteriormente se detalla la adaptación de cada uno de los pasos.

Primero se genera una población inicial P, después, se ingresa al método de descodificación y se evalúa la aptitud de los individuos. Posteriormente se ordenan de forma ascendente en base a su aptitud. Enseguida se clasifica la población en dos conjuntos: El conjunto elite P_e y no elite P_e'. Después se realiza una nueva generación de individuos, en la cual se añade el conjunto de individuos elite. A esta nueva generación se añaden individuos mutantes e individuos generados por el método de cruzamiento. La nueva población se actualiza, y vuelve a entrar a un ciclo hasta que el criterio de parada sea satisfecho. Esta descripción se puede ver en el *Algoritmo 5 basado en BRKGA*.

Algoritmo 5 Basado en BRKGA

$BRKGA(|P|, |P_e|, |P_m|, prob)$

Generación de Población Inicial P

mientras Criterio de parada no sea satifecho **hacer**

Descodificas la población inicial P

Evaluas la aptitud y ordenas P

Clasificas las soluciones en elite P_e y no elite P'_e $(P'_e = P \setminus P_e)$

Actualizas la nueva generación P^+ $(P^+ \leftarrow P^+ \cup P_e)$

Generas mutantes (P_m)

Actualizas la nueva generación P^+ $(P^+ \leftarrow P^+ \cup P_m)$

Generas nuevos individuos con el método de cruzamiento ($P_c, prob$)

Actualizas la nueva generación P^+ $(P^+ \leftarrow P^+ \cup P_c)$

Actualizas la población $P \leftarrow P^+$

fin mientras

El algoritmo tiene los siguientes parámetros de entrada: $|P|$ La cardinalidad de la población inicial, $|P_e|$ el tamaño de la población elite y el tamaño de la población de mutantes. La afinación de estos parámetros se presenta en el capítulo cinco.

4.2.1 Población Inicial

La población inicial es generada de forma aleatoria. Cada vector esta población es llamado cromosoma o llave aleatoria. El tamaño de este conjunto es representado por P.

4.2.2 Codificación

La llave aleatoria R está dividida en tres sub-secuencias $R : \{R_1, R_2, R_3\}$ y tiene un total de alelos $|R|$. Se decidió dividir la llave aleatoria en tres sub-secuencias

debido a la estructura de la solución. Cada alelo r de la primer sub-secuencia R_1 representa al conjunto de lotes de un producto I_n, esta sub-secuencia tiene la misma dimensión que la cantidad de productos existente $|N|$. Como se sabe, estos productos serán programados en ambas sub-líneas de producción, por lo cual, las otras dos sub-secuencias pertenecen a cada una de estas sub-líneas.

La segunda sub-secuencia es subdividida entre el total de etapas $|K_2|$ de la sub-línea de producción de la componente dos. En cada etapa se tienen que programar un total de lotes $|I|$, así en cada una de estas subdivisiones (etapas) se tiene un total de $|I|$ alelos, los cuales, su valor representa la máquina a la cual será asignada cada uno de los lotes. El tamaño de esta sub-secuencia es $|R_2| = |K_2|^*|I|$.

La codificación de la tercer sub-secuencia R_3 es análoga a la segunda, únicamente difiere en lo siguiente: En lugar de tener todas las etapas de la sub-línea de producción uno, quitaremos la última, debido que esta representa la etapa de ensamble y ya es considerada en la segunda sub-secuencia. Por lo cual, el tamaño de esta tercera es $|K_1 - 1|^*|I|$. De esta forma la cantidad de alelos de una llave aleatoria está definida como: $|R| = (|N|) + (|K_2| * |I|) + (|K_1 - 1| * |I|)$

La siguiente figura 4.17 representa una llave aleatoria con un total de tres productos y cinco lotes entre estos tres. La sub-secuencia dos tiene dos etapas, en la primer etapa se tienen tres máquinas y en la segunda etapa se cuenta con dos máquinas. En la sub-secuencia tres sólo hay una etapa con dos máquinas. Como se puede apreciar no se agrega la etapa de ensamble en la tercer sub-secuencia. Los números (1, 2, 3, 4, 5) debajo de cada alelo (valor aleatorio) representan el número del lote al que pertenece.

Sub-secuencia Uno			Sub-secuencia Dos										Sub-secuencia Tres				
Productos			Etapa 1					Etapa De Ensamble					Etapa 1				
0.61	0.32	0.84	0.43	0.15	0.05	0.78	0.99	0.23	0.67	0.74	0.88	0.12	0.57	0.25	0.85	0.68	0.15
			1	2	3	4	5	1	2	3	4	5	1	2	3	4	5

Figura 4.17: Llave aleatoria inicial

En esta sección se llamará un bloque al conjunto de alelos (lotes) juntos de un

mismo producto dentro de una llave aleatoria. En la figura 4.17, un ejemplo de un bloque es la unión de los alelos {0.43, 0.15} los cuales tienen color gris y pertenecen al producto uno.

4.2.3 Descodificación

Previamente a este método, algunos trabajos de la literatura sugieren alternativas para la minimización del makespan, una alternativa es tener la menor cantidad de setup times dentro de las secuencias. Por lo cual, se decidió formar bloques dentro de la segunda y tercer sub-secuencia, así permanecerán juntos los lotes de un mismo producto y únicamente cambiará el orden de los bloques dependiendo del orden de los alelos la primer sub-secuencia. Esto se explica a continuación.

Al inicio, la llave aleatoria no está ordenada. El primer paso consiste en ordenar únicamente la primer sub-secuencia de forma ascendente. De esta forma, el primer conjunto de lotes de un producto a programarse es el que corresponde a aquel alelo que cuenta con el valor aleatorio más pequeño, posteriormente, el siguiente conjunto de lotes a programar son aquellos a los que les corresponde el alelo con el segundo valor aleatorio más pequeño. Este proceso continuará hasta terminar de ordenar los alelos de la primer subsecuencia. El orden establecido en la primer sub-secuencia, prevalecerá en cada una de las etapas (subdivisiones) de las otras dos subsecuencias. Es decir, como se ordenaron los productos, serán ordenados los conjuntos de lotes (bloques) de cada uno de estos en todas las etapas. En la siguiente imagen 4.18 se ilustra lo mencionado.

Sub-secuencia Uno			Sub-secuencia Dos										Sub-secuencia Tres				
Productos			Etapa 1					Etapa De Ensamble					Etapa 1				
0.32	0.61	0.84	0.05	0.78	0.43	0.15	0.99	0.74	0.88	0.23	0.67	0.12	0.85	0.68	0.57	0.25	0.15

Figura 4.18: Llave ordenada

Como se mencionó anteriormente, cada alelo de la segunda y tercera sub-secuencia representan la máquina al que será asignado el lote. En cada una de las

etapas (subdivisiones), se dividirá el intervalo $(0, 1]$ entre el total de máquinas m que haya en estas. Supongamos que hay tres máquinas en una etapa, entonces, el intervalo será dividido en los siguientes sub-intervalos: $(0, 0.333)$, $[0.333, 0.666)$ y $[0.666, 1]$. Si el alelo (lote) perteneciente a esta etapa toma el valor de 0.5, tendrá que asignarse a la máquina dos. Y posteriormente ser secuenciado en base al orden predeterminado por la sub-secuencia uno. Cabe señalar que en la etapa de ensamble los alelos de la sub-secuencia tres están representados por los de la subsecuencia dos, lo anterior para la llegada simultánea a esta etapa.

En la siguiente figura 4.19, debajo de cada alelo (valor aleatorio) de la segunda y tercer subsecuencia se indica la máquina a la que pertenece el alelo (lote), mientras que debajo de cada alelo de la primer sub-secuencia indica el número del producto al que pertenece. En este ejemplo el orden de los productos a programar es primero el producto dos, después el producto uno y al final el producto tres.

Sub-secuencia Uno			Sub-secuencia Dos										Sub-secuencia Tres					
Productos			Etapa 1					Etapa De Ensamble					Etapa 1					
0.32	0.61	0.84	0.05	0.78	0.43	0.15	0.99	0.74	0.88	0.23	0.67	0.12	0.85	0.68	0.57	0.25	0.15	
2	1	3	1	3	2	1	3	2	2	1	2	1	2	2	2	1	1	

Figura 4.19: Llave con asignación de máquina

La siguiente figura 4.20 ilustra cómo el ejemplo anterior es transformado a un arreglo donde se realiza la secuenciación en cada etapa. Es muy importante mencionar que siempre iniciamos con la sub-línea de la componente dos, y posteriormente la sub-línea de la componente uno. Lo anterior es para calcular los tiempos de secuenciación y respetar la llegada simultánea en la etapa de ensamble. Los valores dentro del arreglo representan el número del lote secuenciado, este puede asociarse con la figura 4.17.

La aptitud de esta descodificación será evaluada con la función makespan. Los tiempos de secuencia en las máquinas son calculados de forma similar a la que se hizo en el algoritmo basado en GRASP. Primero se calcula los tiempos en las etapas de la sub-línea de la componente dos, desde la etapa inicial hasta la etapa final y

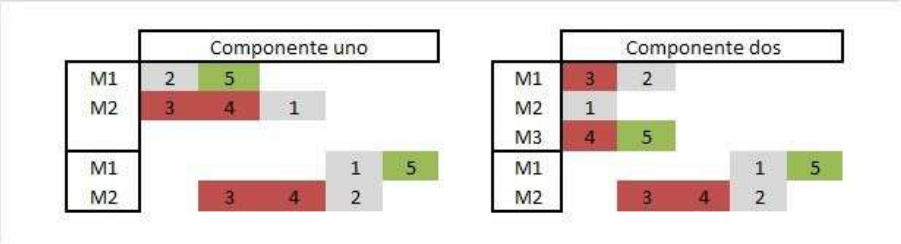


Figura 4.20: Llave transformada

posteriormente, ya se tienen los tiempos de ensamble, se calcula en sentido inverso los de la sub-línea de la componente uno.

4.2.4 Mutantes

En este trabajo se decidió usar el método presentado en Bean et al. [5] para la generación de mutantes. Este operador consiste en crear nuevos cromosomas de forma aleatoria. En el trabajo presentado por Goncalves et al. [21] sugiere que la proporción de estos elementos en cada generación debe ser entre cinco y quince por ciento. El parámetro $|P_m|$ indica la cantidad de elementos mutantes.

4.2.5 Cruzamiento

El operador de cruzamiento que se implementó en este trabajo consiste en seleccionar dos individuos aleatorios, uno es seleccionado del conjunto elite y otro del conjunto no elite. Después de seleccionar los individuos, se genera un número aleatorio entre cero y uno para decidir que alelo de los dos cromosomas prevalecerá en el nuevo individuo. Si el número generado pertenece al intervalo $(0, prob)$ seleccionaremos el alelo del individuo elite, de lo contrario seleccionaremos el alelo del individuo no elite. El tamaño de este conjunto depende del tamaño de los conjuntos elite y mutantes, esto es, $P_c = |P| - |P_e| - |P_m|$.

4.2.6 Criterio de parada

En este trabajo se decidió usar una cantidad total de generaciones. La afinación de este parámetro y los demás, serán presentados en el siguiente capítulo.

4.3 Una metodología basada en Matheuristic

A lo largo del tiempo se han desarrollado diversos algoritmos exactos y metaheurísticos, los cuales han podido resolver problemas de forma exacta o proporcionando buenas soluciones en tiempo razonable. En las últimas décadas, se han intensificado métodos que combinan la programación matemática y metaheurísticas. En el trabajo de Caserta et al. [10], describen la hibridación de métodos exactos y metaheurísticas en términos de una estructura *maestro-esclavo*. En esta estructura puede presentarse los siguientes dos casos:

- El metaheurístico actúa como un *maestro* en un nivel superior y controla al enfoque exacto.
- El método exacto actúa como un *maestro* en un nivel superior y controla al enfoque metaheurístico.

En esta tesis se decidió diseñar una Matheuristic debido a la estructura del modelo matemático. En el modelo presentado en capítulo tres se tienen dos variables de decisión, la asignación y la secuenciación, por lo cual, se buscará descomponer el modelo en dos subproblemas que dependan de cada una de estas dos decisiones.

El algoritmo basado en Matheuristic que se implementó en este trabajo, consiste en descomponer el modelo inicial en dos problemas, a uno se le llamará Problema Principal y al otro Problema Secundario. Estos dos problemas se estarán resolviendo por separado y compartiendo información mediante una restricción que será agregada al Problema Principal basada en el Problema Secundario. El primer problema

realizará un *cluster* de lotes en máquinas, y el segundo realizará la secuenciación dentro de ellas.

Como se mencionó anteriormente, una de las condiciones del problema original es la llegada simultánea de los componentes a la etapa final, este supuesto es muy rígido dentro del modelo inicial. En esta metodología se decidió cambiar el orden de las etapas en sentido inverso. Es decir, la etapa de ensamble será la etapa inicial por la cual tienen que empezar los productos. Posteriormente, pasarán a las demás etapas terminando en la etapa inicial del problema original. De esta forma, se puede decir que primero "se ensambla y después se procesa".

Cabe destacar que en el cambio propuesto, las máquinas siguen conservando los tiempos de procesamiento y los setup times entre lotes se siguen manteniendo. Todos los demás supuestos se conservan. Dicho lo anterior, la solución del problema original no cambia, únicamente se realiza un intercambio entre los tiempos iniciales y finales.

A continuación se presenta y describe el algoritmo de esta metodología. Posteriormente se describe a detalle cada uno de los dos problemas.

Algoritmo 6 Pseudocódigo Matheuristic

$R_{w,m} \leftarrow \emptyset$

$C^*_{max} \leftarrow \infty$

$i \leftarrow 0$

mientras *Criterio de parada* (1) *no sea satisfecho* **hacer**

Resolver el PP : Problema Principal$(R_{w,m})$

$Corte = 0$

mientras *Criterio de parada* (2) *no sea satisfecho* **hacer**

Resolver el PS : Problema Secundario(C^*_{max})

si PS es factible **entonces**

si *Objetivo* $PS > 0$ **entonces**

$R_{w,m} \leftarrow R_{w,m} \cup W_m$:Agregas la secuencia W_m perteneciente a la máquina m en la etapa de ensamble, al conjunto de secuencias $R_{w,m}$

$C^*_{max} \leftarrow C^*_{max} + D * C^*_{max}$:Aumentas el *makespan*

$Corte = 1$ Activas la variable *corte*

si no

$C^*_{max} \leftarrow C^*_{max} - D * C^*_{max}$:Disminuyes el *makespan*

fin si

si no

$C^*_{max} \leftarrow C^*_{max} + D * C^*_{max}$:Aumentas el *makespan*

$Corte = 1$:Activas la variable *corte*

fin si

$i \leftarrow i + 1$

fin mientras

fin mientras

El *Pseudocódigo Matheuristic [6]* consiste en entrar en un ciclo que será realizado mientras el *Criterio de parada* (1) no sea satisfecho. Este criterio de paro es satisfecho si cumple una de las siguientes dos condiciones:

- Realizar un total de iteaciones $Iter$.
- Obtener el mismo valor del parámetro C^*_{max} más de cinco ocasiones en las pasadas iteraciones.

Una vez que se entra al ciclo se resolverá el Problema Principal. Después, se entrará a otro ciclo en el cual se estará resolviendo el Problema Secundario hasta que no sea satisfecho el *Criterio de parada (2)* . Este criterio de paro es satisfecho si cumple una de las siguientes dos condiciones:

- La variable $Corte$ sea igual a uno.
- Encontrar al menos una solución factible en un tiempo límite.

Para continuar con el siguiente ciclo, primero se define el siguiente conjunto: Sea $R_{w,m}$ el conjunto de secuencias w de lotes asignados a la máquina m. Por ejemplo, el conjunto $R_{1,2} : \{3, 4, 6, 10\}$ representa los lotes de la secuencia uno asignados a la máquina dos y se observa que su cardinalidad $|R_{1,2}|$ es cuatro.

Dentro del segundo ciclo, primero se resolverá el Problema Secundario. Si la función objetivo de éste problema es mayor que cero, la secuencia W_m de la máquina m será añadido al conjunto de secuencias $R_{w,m}$, el cual se agregará al Problema Principal para no volver a ser asignados de la misma manera, además, la variable $Corte$ se igualará a uno. De lo contrario, el Problema Secundario se estará resolviendo si la función objetivo es cero y disminuiremos el parámetro C^*_{max} un porcentaje D. Esto para reducir la función objetivo del problema original $Makespan$.

Es importante mencionar, que cuando no se encuentre solución factible, aumentaremos el parámetro C^*_{max} el mismo porcentaje D para volver a la factibilidad, y la variable $Corte$ se igualará a uno. Al finalizar, el algoritmo se quedará con la mejor solución que se haya encontrado antes de salir del primer ciclo.

En este algoritmo se decidio que el parámetro D sea un valor aleatorio dentro del intervalo (0,0.2). Lo anterior es para no tener un mismo aumento o disminución en

el parámetro C^*_{max}. Después de realizar las pruebas computacionales, se observó que antes de las veinte iteraciones el algoritmo convergía, por lo que se decidió el valor del parámetro *Iter* en veinte. En algunos casos no se ejecutaron todas las iteraciones, por lo cual también se decidió la otra opción del *Criterio de parada (1)*, terminar si se repite más de cinco veces el parámetro C^*_{max}. A continuación se describe el Problema Principal y el Problema Secundario.

4.3.1 El Problema principal

El Problema Principal decidirá la asignación de los lotes a las máquinas. Enseguida se presenta y se describe el modelo matemático.

Problema Principal

$$min \sum_{l \in L} \sum_{k \in K^{-1}} C'_{k,l} \tag{4.5}$$

s.t.:

$$\sum_{j \in I} p_{j,m} * y^l_{j,m} \leq C'_{k,l} \qquad m \in M_{k,l}\ k \in K^{-1},\ l \in L \tag{4.6}$$

$$y^l_{i,m} \leq e^l_{i,m} \qquad i \in I,\ m \in M_l,\ l \in L \tag{4.7}$$

$$\sum_{m \in M_{k,l}} e^l_{i,m} * y^l_{i,m} = 1 \qquad i \in I,\ k \in K^{-1},\ l \in L \tag{4.8}$$

$$y^{l_1}_{i,m} - y^{l_2}_{i,m} = 0 \qquad i \in I,\ m \in M_0 \tag{4.9}$$

$$\sum_{i \in R_{w,m}} y^l_{i,m} \leq |R_{w,m}| - 1 \qquad l \in L,\ w \in W,\ m \in M_e \tag{4.10}$$

$$y^l_{i,m} \in \{0,1\} \qquad i \in I,\ m \in M_{k,l}, k \in K,\ l \in L \tag{4.11}$$

$$C'_{k,l} >= 0 \qquad k \in K,\ l \in L \tag{4.12}$$

Los conjuntos siguen siendo los mismos del modelo planteado en el Capítulo Tres, excepto K^{-1} el cual indica las etapas en sentido inverso. La variable de decisión es la asignación de lote i a la máquina m de la componente l: $y^l_{i,m}$. En las

variables $C'_{k,l}$ se guarda el tiempo final de procesamiento dentro de cada etapa. Esta variable se calcula en la restricción (4.6), como se puede apreciar esta restricción no contempla los tiempos de preparación debido a que en este subproblema no se tiene la información de la secuenciación.

El conjunto de restricciones (4.7) - (4.9) contemplan la incompatibilidad y la asignación de lotes a máquinas, así como, la igualdad de lotes del componente uno y componente dos en las máquinas de la etapa inicial M_0. El Problema Secundario busca conectarse con el Problema Principal mediante la restricción (4.10), la cual establece no volver asignar de la misma forma los lotes pertenecientes al conjunto $R_{w,m}$. Para restringir esta asignación, se establece que la sumatoria de las variables de asignación de los lotes pertenecientes a cada secuencia del conjunto $R_{w,m}$ no debe ser mayor a la cardinalidad de cada secuencia ($|R_{w,m}|$) menos uno.

Por último, la función objetivo de este problema (4.5) busca minimizar la suma de los tiempos de procesamiento en totas las máquinas. Esto es para intentar acotar la cantidad de tiempo en cada etapa y buscar soluciones en las cuales se tengan lotes distribuidos en todas las máquinas de cada etapa.

4.3.2 El Problema secundario

El Problema Secundario decide la secuenciación de los lotes en cada máquina y los tiempos de inicio y fin en cada una estas. A continuación se presenta y describe el modelo matemático, es importante señalar que cambia la función objetivo y un conjunto de restricciones.

Problema Secundario

$$min \sum_{i \in I} D_i \tag{4.13}$$

(4.14)

s.t.:

$$x_{i,j}^{m,l} - x_{j,i}^{m,l} \leq 1 \qquad i,j \in I,\ i<j,\ m \in M_l,\ l \in L \qquad (4.15)$$

$$x_{i,j}^{m,l} \leq y_{j,m}^{*l} \qquad i,j \in I,\ i<j,\ m \in M_l,\ l \in L \qquad (4.16)$$

$$\sum_{j \in I} x_{0,j}^{m,l} = 1 \qquad m \in M_{k,l}, k \in K^{-1},\ l \in L \qquad (4.17)$$

$$\sum_{i \in I \cup 0,\, h \neq i} x_{i,h}^{m,l} - \sum_{j \in I \cup 0,\, h \neq j} x_{h,j}^{m,l} = 0 \qquad h \in I,\ m \in M_{k,l}, k \in K^{-1},\ l \in L \qquad (4.18)$$

$$\sum_{j \in I} x_{j,0}^{m,l} = 1 \qquad m \in M_{k,l}, k \in K^{-1},\ l \in L \qquad (4.19)$$

$$\sum_{m \in M_{k,l}} \sum_{i \in I \cup 0} x_{i,j}^{m,l} = 1 \qquad j \in I,\ k \in K^{-1},\ l \in L \qquad (4.20)$$

$$C_{j,k}^{l} \geq S_{i,j}^{k,l} + p_{j,m} * y_{j,m}^{*l} + \tau_{(i,j)}^{m,l} * x_{i,j}^{m,l} - M * (1 - x_{i,j}^{m,l})$$

$$i,j \in I,\ i \neq j,\ m \in M_{k,l}, k \in K^{-1},\ l \in L \qquad (4.21)$$

$$S_{i,j}^{k,l} \geq C_{j,k-1}^{l} \qquad i,j \in I,\ i \neq j,\ k \in K_l,\ s>1,\ l \in L \qquad (4.22)$$

$$S_{i,j}^{k,l} \geq C_{i,k}^{l} \qquad i,j \in I,\ i \neq j,\ k \in K_l,\ l \in L \qquad (4.23)$$

$$x_{i,j}^{m,l_1} - x_{i,j}^{m,l_2} = 0 \qquad i,j \in I,\ i \neq j,\ m \in M_0 \qquad (4.24)$$

$$C_{i,0}^{l_1} - C_{i,0}^{l_2} \leq D_i \qquad i \in I \qquad (4.25)$$

$$C_{i,0}^{l_2} - C_{i,0}^{l_1} \leq D_i \qquad i \in I \qquad (4.26)$$

$$C_{i,n_l}^{l} \leq C_{max}^{*} \qquad i \in I \qquad (4.27)$$

$$x_{i,j}^{m,l}, y_{i,m}^{l} \in \{0,1\} \qquad i,j \in I,\ m \in M_{k,l}, k \in K,\ l \in L \qquad (4.28)$$

$$C_{i,k}^{l}, S_{i,j}^{k,l}, D_i >= 0 \qquad i,j \in I,\ m \in M_{k,l}, k \in K,\ l \in L \qquad (4.29)$$

Después de analizar el modelo inicial se observó que una de las restricciones más fuertes es la igualdad de tiempos de las dos componentes en la etapa de ensamble. Se decidió prescindir de este conjunto de restricciones y cambiarlas por nuevas

restricciones y una función objetivo que ayuden a beneficiar la llegada simultánea en esta etapa.

El conjunto de restricciones (4.15) - (4.24) son iguales a las del modelo planteado en el capítulo tres, únicamente difieren en que en este conjunto existe una variable $y^{*l}_{j,m}$ la cual ya es conocida después de resolver el Problema Principal. Las desigualdades (4.25)-(4.26) indican la diferencia D entre los tiempos finales de cada pareja de lotes de la componente uno y dos en la etapa de ensamble, ahora etapa inicial. En este problema usaremos un parámetro que irá disminuyendo de acuerdo al algoritmo, este parámetro es el C^*_{max}. En la restricción (4.27), este parámetro acota el tiempo final del último lote en la última etapa, antes etapa inicial.

La función objetivo busca minimizar la sumatoria de las diferencias de los tiempos en la etapa de ensamble. De esta forma se buscara que esta diferencia tienda a ser cero para lograr la llegada simultánea. Las variables D son mayores o iguales a cero.

Capítulo 5

Experimentación computacional y resultados

Este capítulo describe el ambiente computacional con el que fue realizada la experimentación. Además, describe las instancias y la forma en que fueron generadas. Así también, muestra los resultados que se obtuvieron de la afinación de parámetros. Por último, detalla las métricas de evaluación y los resultados finales de las instancias.

5.1 Ambiente computacional e instancias

En esta sección primero se presenta como se desarrolló la generación de instancias y posteriormente se muestran las características del equipo de cómputo en el que fue desarrollada la investigación.

Para esta tesis, las instancias fueron creadas debido a que no existe en la literatura instancias con las mismas caracteristicas del problema aquí estudiado. Además, las instancias fueron clasificadas en cuatro tipos. Cada una de estas clasificaciones cuenta con cinco instancias para un total de veinte. El tamaño de cada tipo de instancia depende del total de productos, total de lotes, número de etapas y número de máquinas.

En el problema real, el total de lotes depende de tres factores, la demanda (d_i), el safety stock (ss_i) y el inventario (h_i) de cada producto, estos tres son conocidos por parte de la empresa. Para encontrar el total de lotes por producto, primero sumamos la demanda y el safety stock y le restamos lo que hay en inventario ($d_i + ss_i - h_i$) y posteriormente los dividimos entre veinticinco para saber cuántos lotes hay. Los lotes no necesariamente tienen que tener el mismo tamaño. A continuación se muestra la Tabla 5.1, la cual contiene los intervalos de los parámetros descritos anteriormente.

Tabla 5.1: Parámetros de pre-procesamiento

Parámetro	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Demanda	40-65	50-100	50-200	50-300
Safety Stock	10-15	20-50	40-70	60-90
Inventario	5-10	15-40	25-50	35-60

De cada uno de los intervalos de la Tabla 5.1 fue seleccionado de forma aleatoria un valor para los parámetros de Demanda, Safety Stock e Inventario. Los límites inferiores y superiores de esta tabla fueron sugeridos por parte de la empresa de acuerdo al tipo de instancia. Las instancias de Tipo 1 representan a las de tamaño más pequeño, por lo cual, tienen los límites menores de la tabla. Así como, las instancias de tipo cuatro representan las de tamaño más grande y por ende, cuentan con los límites mayores de la tabla.

A continuación se presenta la Tabla 5.2, en la cual se representan los intervalos de todos los parámetros utilizados para la generación de instancias.

Tabla 5.2: Parámetros

Tipo	Prod.	Lotes	Eta. C1	Eta. C2	Máq. C1	Máq. C2	Cap. M/h C1	Cap. M/h C2	Tiem. de prep.
1	4	10-12	2	2	4	3	40-70	10-20	25-45
2	10	29-35	3	3	5	5	40-70	15-25	25-45
3	30	50-60	5	6	10	12	60-90	20-30	25-45
4	50	90-125	12	14	25	29	40-65	10-20	25-45

La columna uno de la Tabla 5.2 representa el tipo de instancia, mientras que las columnas dos y tres indican la cantidad de productos y el intervalo del total de lotes que hay en cada tipo de instancia. Las columnas cuatro y cinco muestran la cantidad de etapas en las sub-líneas de la componente uno (C1) y la componente dos (C2). Las columnas seis y siete representan la cantidad de máquinas de C1 y C2 respectivamente. Por último, las columnas ocho y nueve, muestran los intervalos de las capacidades de las máquinas por hora, de los cuales fue seleccionado un valor aleatorio para cada etapa, es decir cada máquina tiene el mismo valor de la capacidad. Así como, en la última columna representa el intervalo de los tiempos de preparación del que también fue seleccionado un valor aleatorio para cada máquina en cada instancia. Las instancias de tipo cuatro emulan el comportamiento del caso real de la empresa.

Las características del equipo de cómputo donde se desarrollaron las pruebas computacionales son: un procesador intel core (R) con una velocidad de 2.9 GHz y 8 gb de memoria ram. Los códigos fueron implementados en Visual Studio 2012 en lenguaje de programación C++. Los modelos matemáticos fueron implementados en Ilog Cplex 12.6.

5.2 Afinación de parámetros

En este trabajo se utilizó freeware calibra para la afinación de los parámetros de los algoritmos basados en GRASP y BRKGA. El calibra fue presentado por Adenso-Díaz y Laguna [1] en el año de 2001. Calibra esta basado en la combinacion de un diseño de experimentos y una búsqueda local. El diseño de experimentos proporciona una manera de enfocar la búsqueda en los espacios de búsqueda más prometedores. A continuación se muestra la Tabla 5.3 la cual contiene los resultados de dichos parámetros.

Tabla 5.3: Afinación de parámetros

Heurístico basado en GRASP			Heurístico basado en BRKGA		
Parámetro	Rango	Valor	Parámetro	Rango	Valor
α	0-1	0.3	$\|P\|$	100-1000	887
$Iter$	100-1000	763	$\|P_m\|$	0- 0.3	0.3
			$\|P_e\|$	0- 0.7	0.6
			p_c	0.5-1	0.9
			Tot_{gen}	10-100	78

Los valores que fueron afinados en el algoritmo tipo Grasp son: la aleatoriedad de la lista de candidatos α y el total de iteraciones $Iter$. Mientras que en el BRKGA los parámetros fueron el total de la población $|P|$, la proporción de mutantes $|P_m|$, la proporción del conjunto elite $|P_e|$, la probabilidad de cruce p_c y el total de generaciones Tot_{gen}. Estos últimos valores encontrados son muy similares a los propuestos en la literatura de estas metaheurísticas.

5.3 Resultados

En esta sección se presentan los resultados finales de la implementación del modelo matemático resuelto en CPLEX y los tres algoritmos presentados en el capítulo

cuatro. Cada uno de estos métodos fueron probados con las instancias presentadas en la sección anterior.

En este trabajo se decidieron utilizar tres métricas para evaluar el desempeño de los tres algoritmos. La primer métrica consiste en encontrar el mejor valor de la función objetivo después de finalizar el algoritmo. Debido a que en los tres algoritmos se realizan un conjunto de iteraciones, se decidió que la segunda métrica fuese el promedio de los valores de las funciones objetivos encontrados en todas las iteraciones. Por último, la tercer métrica muestra el tiempo promedio que se tardó en correr cada instancia.

A continuación se presenta la tabla 5.4, en ella se muestran los resultados de la métrica uno (mejor valor de la función objetivo).

Tabla 5.4: Resultados métrica 1

Instancia	CPLEX	GAP	Algoritmo GRASP	Algoritmo BRKGA	Matheuristic
1	1365.82	0.68	1420.36	1402.95	**1389.1**
2	891.636	0.24	949.81	881.636	**880.92**
3	1297.93	0.73	1320.33	1300.24	**1289.89**
4	1106.77	0.68	1200.54	1154.38	**1132.66**
5	1194	0.70	1229	1212	**1197.25**
6	-	-	**4775.29**	4778.14	6589.08
7	3411.88	0.97	2779.06	**2749.88**	4693.73
8	4722.18	1	**4061.01**	4062.18	5131.76
9	4180.94	1	3674.52	**3665.95**	5368.2
10	-	-	4852.75	**4824.25**	6388.94
11	-	-	**3557.48**	3638.22	3754.34
12	-	-	**3738.06**	3805.73	5310.25
13	-	-	**3328.96**	3457.52	3590.62
14	-	-	**2990**	3047.87	5271.04
15	-	-	**3983.19**	3985.17	4037.58
16	-	-	**12640.4**	14241.4	33730.5
17	-	-	**11087.2**	14103.2	-
18	-	-	**12769.2**	16528.4	-
19	-	-	**14784.7**	22467.8	-
20	-	-	**11496.7**	14443.6	-

La primer columna representa la instancia, la segunda columna representa el valor de la función objetivo encontrado por CPLEX y a un lado se encuentra el valor del GAP encontrado por CPLEX.

La cuarta, quinta y sexta columna representan el mejor valor de la función objetivo encontrado por los algoritmos GRASP, BRKGA y Matheuristic respecti-

vamente. Como podemos observar, el optimizador CPLEX solo encontró soluciones en las instancias del uno al cinco, y del siete al nueve. Lo anterior en un límite de tiempo de una hora. Si comparamos únicamente las tres metodologías planteadas en el capítulo cuatro, podemos ver el valor más pequeño de cada instancia en ***letra negrita***.

En la Tabla 5.5, se calculó el GAP (multiplicado por cien) de las tres metodologías implementadas en este trabajo. Donde el mejor valor está mostrado en ***letra negrita***. A continuación se presentan los resultados de la Tabla 5.5:

Tabla 5.5: Gap del mejor valor encontrado

Instancia	Algoritmo GRASP	Algoritmo BRKGA	Matheuristic
1	2.27 %	1.00 %	**0.00 %**
2	7.82 %	0.08 %	**0.00 %**
3	2.36 %	0.80 %	**0.00 %**
4	5.99 %	1.92 %	**0.00 %**
5	2.65 %	1.23 %	**0.00 %**
6	**0.00 %**	0.06 %	37.98 %
7	1.06 %	**0.00 %**	70.69 %
8	**0.00 %**	0.03 %	26.37 %
9	0.23 %	**0.00 %**	46.43 %
10	0.59 %	**0.00 %**	32.43 %
11	**0.00 %**	2.27 %	5.53 %
12	**0.00 %**	1.81 %	42.06 %
13	**0.00 %**	3.86 %	7.86 %
14	**0.00 %**	1.94 %	76.29 %
15	**0.00 %**	0.05 %	1.37 %
16	**0.00 %**	12.67 %	166.85 %
17	**0.00 %**	27.20 %	-
18	**0.00 %**	29.44 %	-
19	**0.00 %**	51.97 %	-
20	**0.00 %**	25.63 %	-

En esta Tabla 5.5, si el valor del GAP de una metodología es cero por ciento, quiere decir que la metodología obtuvo el mejor valor del GAP. Por otra parte, la Tabla 5.6 muestra los resultados de la segunda métrica (promedios) evaluando las tres metodologías.

Tabla 5.6: Resultados métrica 2

Instancia	GRASP	GAP (GR)	BRKGA	GAP (BR)	Matheuristic	GAP (MA)
1	1420.37	0.01 %	1411.6	0.62 %	**1396.33**	0.52 %
2	1033.2	8.78 %	929.13	5.39 %	**885.42**	0.51 %
3	1320.32	0.01 %	1308.25	0.62 %	**1300.95**	0.86 %
4	1200.54	0.00 %	1165.10	0.93 %	**1137.28**	0.41 %
5	1236.19	0.59 %	1218.52	0.54 %	**1213.20**	1.33 %
6	4826.73	1.82 %	**4799.21**	6.33 %	6698.52	6.90 %
7	**2829.55**	1.08 %	2924.02	0.44 %	5017.54	1.66 %
8	4067.12	0.15 %	**4067.12**	0.12 %	5205.09	1.43 %
9	3700.34	0.70 %	**3678.77**	0.35 %	5368.2	0.00 %
10	4863.15	0.21 %	**4835.29**	0.23 %	6408.29	0.30 %
11	**3667.81**	7.67 %	3917.23	3.10 %	3754.34	0.00 %
12	**4199.97**	12.36 %	4228.37	11.11 %	5310.25	0.00 %
13	3857.13	15.87 %	3964.69	14.67 %	**3590.62**	0.00 %
14	**3229.98**	8.03 %	3309.18	8.57 %	5271.04	0.00 %
15	4199.87	5.44 %	4228.37	6.10 %	**4037.58**	0.00 %
16	**14201.2**	12.35 %	18393.7	29.16 %	33730.5	0.00 %
17	**12486.8**	12.62 %	20660.2	46.49 %	-	-
18	**14377.9**	12.60 %	25785.4	56.01	-	-
19	**16095.6**	8.87 %	45194.9	101.15	-	-
20	**12854.1**	11.81 %	19757.2	36.79 %	-	-

La métrica dos (promedio) mide la eficiencia del conjunto de soluciones generadas en todas las iteraciones de cada algoritmo. La primer columna nos indica la instancia. Las columnas pares (2, 4, 6), representan el promedio de la función objetivo de todas las iteraciones. Mientras que las columnas impares (3, 5 y 7), muestran el GAP comparando el valor promedio y el mejor valor de la función objetivo encon-

trado en cada algoritmo.

El GAP presentado en la Tabla 5.5, ayuda a mostrar la cercanía en la función objetivo de las soluciones encontradas en cada algoritmo. Valores muy cercanos a cero por ciento, indican que las soluciones fueron muy cercana a la mejor solución en su respectiva instancia. Por último, el mejor valor promedio de cada instancia comparando todos los algoritmos es resaltado en **letra negrita**.

A continuación se presenta la Tabla 5.7, en ella se presenta la Métrica tres, que muestra los tiempos en segundos de cada algoritmo en cada instancia.

Tabla 5.7: Resultados métrica 3

Instancia	GRASP	BRKGA	Matheuristic
1	1	< 1	2117
2	< 1	1	1176
3	< 1	1	2287
4	< 1	< 1	2227
5	< 1	< 1	2241
6	**1**	**1**	8145
7	**1**	8	7645
8	**2**	1	7294
9	**1**	4	7585
10	**2**	**2**	8121
11	**15**	24	7958
12	**20**	28	13225
13	**15**	23	9243
14	**12**	22	7117
15	**15**	24	7659
16	293	**102**	34940
17	319	**107**	40300
18	338	**110**	45242
19	483	**134**	-
20	360	**113**	-

Como se puede apreciar las heurísticas basadas en GRASP y BRKGA obtuvieron tiempos muy bajos en la mayoría de las instancias. Los valores en negritas son los de menor tiempo En el siguiente capítulo se harán las conclusiones basadas en el desempeño de las tres métricas presentadas en este capítulo.

Capítulo 6

Conclusiones y trabajo a futuro

En este trabajo se estudió un problema de programación de producción en una línea de ensamble. La línea de producción consta de dos sub-líneas de producción las cuales son unidas en una etapa de ensamble. En cada sub-línea se tienen un conjunto de etapas por las cuales tienen que pasar todos los lotes de producción. En cada etapa se tiene máquinas en paralelo. El flujo de producción es unidireccional. Se busca minimizar el makespan en la última etapa de la línea de producción. Este problema puede verse como una variante del Hybrid Flow Shop con ensamble.

Primero, se representó el problema con un modelo matemático, el cual fue descrito a detalle en el capítulo tres. Este modelo fué implementado en el lenguaje de programación C++ y resuelto en el optimizador CPLEX. Posteriormente, después de analizar los resultados presentados y analizar la complejidad del problema (NP-duro), se decidió implementar un conjunto de metodologías para obtener soluciones de buena calidad.

Dentro de la revisión de literatura no se encontró un problema que posea la integración de todas las características mostradas en el capítulo tres. Existen algunos trabajos con similitudes, pero no hay uno que aborde la misma problemática.

6.1 Conclusiones generales

En esta sección se describen las conclusiones del problema, del modelo matemático y las tres metodologías implementadas.

La integración de las características como la incompatibilidad y la llegada simultánea de los lotes en la etapa de ensamble en un HFS, son las variantes que presenta este problema comparado con los trabajos de la literatura. Por lo cual, el modelo matemático presentado en el Capítulo tres, es uno de los aportes de este trabajo a la literatura. En este trabajo se presentó una formulación clásica con variables de asignación, de secuenciación (inmediata) y de tiempo. Sin embargo, la diferencia consiste en la integración de todas las consideraciones planteadas dentro de las restricciones.

De acuerdo a los resultados presentados en la Tabla 5.4, observamos que la implementación en CPLEX solo resuelve ocho instancias de un total de veinte en un tiempo de una hora. Las instancias del tipo uno se obtuvieron GAPS entre veinticinco y setenta por ciento, lo cual implica que el problema no puede ser resuelto de forma exacta en un tiempo razonable. Debido a la complejidad del problema (NP-duro) y a los resultados mostrados en Capítulo tres se decidió optar por otro tipo de metodologías. A continuación se presentan las conclusiones de las metodologías implementadas en este trabajo.

De acuerdo a la Tabla 5.5 la metodología basada en GRASP obtuvo un total de once valores mínimos. Siendo la mayor cantidad de estos valores en las instancias de tipo tres y tipo cuatro. Además, podemos decir que los valores obtenidos por la Metodología GRASP implican que las soluciones no fueron encontradas de forma casual, esto debido a los valores mostrados en la Tabla 5.6, indican que el gap de los promedios no rebasa del quince por ciento en todas las instancias. Así también, la Tabla de tiempos 5.7, se reporta que en las primeras quince instancias (tipo uno, dos y tres) se obtuvó un tiempo de ejecución menor de quince segundos. A excepción de

las instancias tipo cuatro, las cuales poseen tiempos entre trescientos y cuatrocientos segundos.

La metodología basada en BRKGA ganó en tres instancias de un total de veinte en la métrica del mejor valor de la función objetivo. De acuerdo a la Tabla 5.5, se obtuvieron GAP's menores de un cuatro por ciento en las primeras quince instancias. Los cuáles representan porcentaje muy cercanos a los mejores valores. Así como, se obtuvieron tiempos competitivos comparados con la metodología GRASP en las instancias tipo uno, dos y tres, mientras que en las instancias tipo cuatro se encontraron los mejores tiempos de las tres metodologías. Sin embargo sus valores objetivos no son buenos en las instancias tipo cuatro comparado con las otras dos.

La metodología basada en Matheuristic obtuvo cinco mejores valores en la métrica uno. Sus promedios son muy cercanos al mejor valor encontrado en todas las iteraciones. Por lo cual, podemos decir que este algoritmo converge muy rápido. Los tiempos requeridos por esta metodología son muy grandes. Se sugiere la implementación de este algoritmo en instancias pequeñas.

Por último, podemos concluir que las metodología basada en GRASP es mejor en calidad de la función objetivo comparada con las otras dos. Por lo cual, se sugiere la implementación de esta metodología en el caso real. Sin embargo, existen diversas líneas de producción que tienen menos etapas y poseen carcaterísticas muy similares. En esos casos se puede implementar la metodología del l BRKGA. En empresas donde se tenga alguna licencia de un optimizador y tenga una línea de producción no mayor a tres etapas, se puede implementar el algoritmo basado en Matheurístic.

6.2 Trabajo a futuro

En esta sección se presentan las diferentes extensiones que pueden existir de ésta tesis.

Como primera extensión, se pretende incorporar en el modelo matemático el parámetro de tiempos de entrega para cada tarea. Compañías dedicadas a la elaboración de chips o dispositivos electronicos, son casos de líneas de producción de ensamble donde se presentan periodos cortos de entrega.

Por otra parte, una vez incorporado el parámetro de las fechas de entrega, se puede considerar un segundo objetivo. Este objetivo se pretende que sea minimizar la tardanza. Esto con el fin de encontrar un equilibrio que beneficie al cliente y a la empresa al mismo tiempo.

En este mismo contexto de programación multiobjetivo, la reprogramación es un tema que se puede considerar. Una vez que se tiene una decisión de la programación de producción, diversas causas pueden ser propensas a modificar esta decisión, por lo cual se pretende considerar un segundo objetivo que minimice la cantidad de movimiento de una solución inicial sujeto a todas las restricciones ya establecidas junto con su objetivo inicial.

Otro parámetro importante es la incorporación de capacidades en los "buffers". En este trabajo, la empresa contaba con amplias lugares de almacenamiento entre estaciones de trabajo, por lo cual no se consideró dentro de la tesis. Sin embargo, sabemos que no todas las empresas cuentan con una amplia infraestructura, por lo que sería conveniente analizar los cambios pertinentes en el modelo con la incorporación de éste parámetro.

En la actualidad, diversos trabajos estudian la planeación y programación en conjunto. Como extensión de este trabajo se contempla el decidir cuantos elementos de cada lote tienen que programarse y a su vez decidir la programación de producción. De acuerdo a algunos modelos presentados en la literatura, (Ibarra-Rojas et al. [33]) podemos considerar que la integración de esta decisión a nuestro modelo, lo volvería un problema no lineal. Por lo cual, sería muy interesante aplicar un algoritmo basado en la Matheuristic como el que se presentó en este trabajo.

Por último, una de las consideraciones más fuertes de este problema son las

llegadas simultáneas en la etapa de ensamble. Una extensión importante sería considerar una tolerancia ϵ mayor que cero. Este parámetro podría ayudar significativamente a soluciones más flexibles de este problema.

Bibliografía

[1] Adenso-Diaz, B. y M. Laguna, «Fine-tunning of algorithm using fractional experimental designs and local search», *Operations Research*, **54**(1), págs. 99–114, 2006.

[2] Agnetis, A., A. Pacifici, F. Rossi, M. Lucertini, S. Nicoletti, F. Nicolo, G. Oriolo, D. Pacciarelli y E. Pesaro, «Scheduling of flexible flow lines in an automobile assembly plant», *International Journal of Production Economics*, **97**(2), págs. 348–362, 1997.

[3] Alharkan, I. M., *Algorithm for sequencing and scheduling*, primera edición, Springer Science and Business Media, USA, 1997.

[4] Allahverdi, F., A. & Al-Anzi, «Scheduling multi-stage parallel processor services to minimize average response time», *Journal of the Operational Research Society*, **57**(1), págs. 101–110, 2006.

[5] Bean, J. C., «Genetic algorithms and random keys for sequencing and optimization», *ORSA Journal on Computing*, **6**(1), págs. 154–160, 1994.

[6] Beaulieu, H., J. A. Ferland, B. Gendron y P. Michelon, «A mathematical programming approach for scheduling physicians in the emergency room», *Health Care Management Science*, **3**(1), págs. 193–200, 2000.

[7] Behnamian, J. y M. Zandieh, «A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties», *Expert Systems with Applications*, **38**(12), págs. 14 490–14 498, 2011.

[8] BERTEL, S. y J.-C. BILLAUT, «A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation», *European Journal of Operational Research*, **159**(3), págs. 651–662, 2004.

[9] BRAH, S. A. y J. L. HUNSUCKER, «Branch and bound algorithm for the flow shop with multiple processors», *European Journal of Operational Research*, **51**(1), págs. 88–99, 1991.

[10] CASERTA, M. y S. VOSS, «Metaheuristics: Intelligent problem solving.», *Matheuristics, Annals of Information Systems*, **10**(1), págs. 1–38, 2009.

[11] CHEN, L., N. BOSTEL, P. DEJAX, J. CAI y L. XI, «A tabu search algorithm for the integrated scheduling problem of container handling systems in a maritime terminal», *European Journal of Operational*, **181**(1), págs. 40–58, 2007.

[12] CHOI, H.-S., J.-S. KIM y D.-H. LEE, «Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line», *Expert Systems with Applications*, **38**(1), págs. 3514–3521, 2011.

[13] DOMÍNGUEZ MACHUCA, J. A., *Dirección de Operaciones: Aspectos estratégicos en la Producción y los servicios*, primera edición, Editorial McGraw–Hill, Madrid, 1995.

[14] FATTAHI, P., S. HOSSEINI y F. JOLAI, «Some heuristics for the hybrid flow shop scheduling problem with setup and assembly operations», *International Journal of Industrial Engineering Computations*, **4**(1), págs. 393–416, 2013.

[15] FATTAHI, P., S. M. H. HOSSEINI y F. JOLAI, «A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem», *Int J Adv Manuf Tech*, **65**(1), págs. 787–802, 2013.

[16] FATTAHI, P., S. M. H. HOSSEINI, F. JOLAI y R. TAVAKKOLI-MOGHADDAM, «A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations», *Applied Mathematical Modelling*, **38**(1), págs. 119–137, 2014.

[17] FEO, T. A. y M. G. RESENDE, «A probabilistic heuristic for a computationally difficult set covering problems», *Operations Research Letters*, **8**(2), págs. 109–133, 1989.

[18] GAITHER, N. y G. FRAZIER, *Administración de Producción y Operaciones*, primera edición, Internacional Thomson Editoresl, México, 2000.

[19] GICQUEL, C., L. HEGE, M. MINOUX y W. VAN CANNEYT, «A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-wait constraints», *Computers & Operations Research*, **39**(3), págs. 629–636, 2012.

[20] GÓMEZ-GASQUET, P., «Programación de la producción en un taller de flujo híbrido sujeto a incertidumbre: arquitectura y algoritmos. Aplicación a la industria cerámica», Departamento de Organización de Empresas. Universidad Politécnia de Valencia, 2010.

[21] GONÇALVES, J. F. y M. G. RESENDE, «Biased random-key genetic algorithms for combinatorial optimization», *Journal Heuristics*, **17**(5), págs. 487–525, 2011.

[22] GONÇALVES, J. F., M. G. RESENDE y J. J. MENDES, «A Biased Random-Key Genetic Algorithm with Forward-Backward Improvement for the Resource Constrained Project Scheduling Problem», .

[23] GRABOWSKI, J. y J. PEMPERA, «Sequencing of jobs in some production system», *European Journal of Operational Research*, **125**(3), págs. 535–550, 2000.

[24] GRAHAM, R. L., E. L. LAWLER, J. K. LENSTRA y A. R. KAN, «Optimization and approximation in deterministic sequencing and scheduling: a survey», *Annals of Discrete Mathermatics*, **5**(1), págs. 287–326, 1977.

[25] GUINET, A. y M. SOLOMON, «Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time», *International Journal of Production Research*, **34**(6), pág. 1643–1654, 2011.

[26] GUPTA, J. N., «Two-Stage, Hybrid Flowshop Scheduling Problem», *Operations Research Society*, **39**(4), págs. 359–364, 1988.

[27] GUPTA, J. N. y E. A. TUNC, «Schedules for a two-stage hybrid flow shop with parallel machines at the second stage», *The International Journal of Production Research*, **29**(7), págs. 1489–1502, 1991.

[28] HAOUARI, M., L. HIDRI y A. GHARBI, «Optimal scheduling of a two-stage hybrid flow shop», *Mathematical Methods of Operations Research*, **64**(1), págs. 107–124, 2006.

[29] HE, D. y A. BABAYAN, «Scheduling manufacturing systems for delayed product differentiation in agile manufacturing», *International Journal of Production Research*, **40**(11), págs. 2461–2481, 2002.

[30] HERRERA, M., «Programación de la producción en una empresa automotriz», Tesis de Maestría, Posgrado en Ingeniería en Sistemas, Universidad Autonoma de Nuevo León, 2016.

[31] HOLLAND, J. H., *Adaptation in Natural and Artificial Systems*, primera edición, University of Michigan Press, Ann Arbor, 1975.

[32] IBARRA-ROJAS, O. y Y. RIOS-SOLIS, «Diseño de horarios para sincronizar líneas de buses en múltiples periodos de planificación», *INGENIERÍA DE TRANSPORTE*, **18**(1), págs. 24–31, 2013.

[33] IBARRA-ROJAS, O. J., R. Z. RÍOS-MERCADO, Y. A. RIOS-SOLIS y M. A. SAUCEDO-ESPINOSA, «A decomposition approach for the piece–mold–machine manufacturing problem», *International Journal of Production Economics*, **134**(1), págs. 255–261, 2013.

[34] JANIAK, A., E. KOZAN, M. LICHTENSTEIN y C. OĞUZ, «Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion», *International Journal of Production Economics*, **105**(2), págs. 407–424, 2007.

[35] JOHNSON, S. M., «Optimal two and three stage production schedules with set up times included», *Naval Research Logistic Quarterly*, **1**(1), págs. 61–68, 1954.

[36] KURZ, M. E. y R. G. ASKIN, «Comparing scheduling rules for flexible flow lines», *International Journal of Production Economics*, **85**(3), págs. 371–388, 2003.

[37] LEE, C.-Y. y G. L. VAIRAKTARAKIS, «Performance comparison of some classes of flexible flow shops and job shops», *The International Journal of Flexible Manufacturing Systems*, **98**(1), págs. 379–405, 1998.

[38] LIAO, C.-J., E. TJANDRADJAJA y T.-P. CHUNG, «An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem», *Expert Systems with Applications*, **12**(6), págs. 1755–1764, 2012.

[39] LIN, H.-T. y C.-J. LIAO, «A case study in a two-stage hybrid flow shop with setup time and dedicated machines», *International Journal of Production Economics*, **86**(2), págs. 133–143, 2003.

[40] LOGENDRAN, R., F. BARNARD *et al.*, «Sequence-dependent group scheduling problems in flexible flow shops», *International Journal of Production Economics*, **102**(1), págs. 66–86, 2001.

[41] PINEDO, M., *Planning and Scheduling in Manufacturing and Services*, primera edición, Springer Science and Business Media, USA, 2005.

[42] PINEDO, M. L., *Scheduling: Theory, Algorithms, and Systems*, Springer Science & Business Media, 2012.

[43] REBAINE, D., «Scheduling flexible flowshops with unit-time operations and minimum time delays», *Electronic Notes in Discrete Mathematics*, **36**(1), págs. 1193–1200, 2010.

[44] RESENDE, M. G. y J. L. GONZÁLEZ-VELARDE, «GRASP: Greedy Randomized Adaptive Search Procedures», *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial*, **7**(2), págs. 67–71, 2003.

[45] RUIZ, R. y C. MAROTO, «A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility», *European Journal of Operational Research*, **169**(3), págs. 781–800, 2006.

[46] RUIZ, R. y J. A. VÁZQUEZ-RODRÍGUEZ, «The hybrid flow shop scheduling problem», *European Journal of Operational Research*, **205**(12), págs. 1–18, 2010.

[47] SAWIK, T., A. SCHALLER y T. M. TIRPAK, «Scheduling of printed wiring board assembly in Surface mount technology lines», *Journal of Electronics Manufacturing*, **11**(1), págs. 1–17, 2002.

[48] SUNG, C. y H. A. KIM, «A two-stage multiple-machine assembly scheduling problem for minimizing sum of completion times», *Int. J. Production Economics*, **113**(1), págs. 1038–1048, 2008.

[49] TAVAKKOLI-MOGHADDAM, R., N. SAFAEI y F. SASSANI, «A memetic algorithm for the flexible flow line scheduling problem with processor blocking», *Computers and Operations Research*, **36**(2), págs. 402–414, 2009.

[50] TORABI, S. A., S. F. GHOMI y B. KARIMI, «A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains», *European Journal of Operational Research*, **173**(1), págs. 173–189, 2006.

[51] TSUBONE, H., M. OHBA, H. TAKAMUKI y Y. MIYAKE, «Production scheduling system for a hybrid flow shop a case study», *Omega*, **21**(2), págs. 205–214, 1993.

[52] VOSS, S. y A. WITT, «Hybrid flow shop scheduling as a multi mode multiproject scheduling problem with batching requirements: A real world application», *International journal of production economics*, **105**(2), págs. 445–458, 2007.

[53] ZIJM, W. y E. NELISSEN, «Scheduling a flexible machining centre», *Engineering Costs and Production Economics*, **19**(1), págs. 249–258, 1990.

RESUMEN AUTOBIOGRÁFICO

Rafael Muñoz Sánchez

Candidato para obtener el grado de

Doctor en Ingeniería con
Especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis

TÉCNICAS DE OPTIMIZACIÓN PARA RESOLVER UN PROBLEMA DE PROGRAMACIÓN DE PRODUCCIÓN DE UNA LÍNEA DE ENSAMBLE

Nací el 19 de septiembre de 1989 en Tampico, Tamaulipas. Realicé mis estudios de licenciatura en matemáticas y maestría en ciencias con orientación en matemáticas en la Facultad de Ciencias Físico Matemáticas de la Universidad Autónoma de Nuevo León. Ingresé en el 2014 al doctorado en Ingeniería con especialidad en ingeniería de sistemas, de la FIME, bajo la dirección de la Dra. Iris Abril. He sido profesor de matemáticas en la UANL y el TecMilenio. Actualmente trabajo en el área de modelación matemática en una empresa de consultoría en optimización.