BALANCEO DE LINEAS DE ENSAMBLE DE MODELO MIXTO MEDIANTE ENCADENAMIENTO DE HEURISTICAS

T E S I S

MAESTRIA EN CIENCIAS EN SISTEMAS INTELIGENTES

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Por

ING. HALLEY JARUMY FERRER SORIANO

DICIEMBRE DE 2009

BALANCEO DE LINEAS DE ENSAMBLE DE MODELO MIXTO MEDIANTE ENCADENAMIENTO DE HEURISTICAS

T E S I S

MAESTRIA EN CIENCIAS EN SISTEMAS INTELIGENTES

INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

Por

ING. HALLEY JARUMY FERRER SORIANO

DICIEMBRE DE 2009

Balanceo de Líneas de Ensamble de Modelo Mixto Mediante Encadenamiento de Heurísticas

T E S I S

Maestría en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. Halley Jarumy Ferrer Soriano

Diciembre 2009

Balanceo de Líneas de Ensamble de Modelo Mixto Mediante Encadenamiento de Heurísticas

TESIS

Maestría en Ciencias en

Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. Halley Jarumy Ferrer Soriano

Diciembre 2009

Balanceo de Líneas de Ensamble de Modelo Mixto Mediante Encadenamiento de Heurísticas

Por

Ing. Halley Jarumy Ferrer Soriano

TESIS

Presentada a la División de Mecatrónica y Tecnologías de Información

Este trabajo es requisito parcial para obtener el grado académico de Maestro en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Monterrey, N.L. Diciembre de 2009

A mi familia y amigos que creyeron en mi y me dieron su apoyo, en especial a la memoria de mi Madre Guadalupe Soriano Juárez (Q.E.P.D.) porque fue el pilar fundamental de mi vida, por todo su amor y apoyo, en resumen porque mi sueño fue su sueño.

Reconocimientos

Deseo externar un sincero agradecimiento a las personas que de alguna forma colaboraron a la realización de esta investigación de tesis.

Al Dr. Santiago Enrique Conant Pablos, por su dirección y ayuda constante durante el desarrollo de esta investigación, en especial por su orientación metodológica y por su contínuo estímulo durante todo el proceso hasta al final del mismo.

A mis sinodales, Dr. Hugo Terashima Marín y Dr. Eduardo Uresti Charre, por su tiempo, aportaciones y dedicación en la revisión de tesis.

A mis profesores de la maestría, por su enseñanza y por formar parte de mi triunfo académico.

A mis padres y familia en especial a mis primas Brígida y Eufemia Cruz Soriano, y Antonieta Soriano Cruz por brindarme todo su amor y apoyo para triunfar en esta etapa.

A mis compañeros y amigos quienes han compartido conmigo este proceso y que siempre me dieron ánimos en los momentos difíciles.

HALLEY JARUMY FERRER SORIANO

Instituto Tecnológico y de Estudios Superiores de Monterrey
Diciembre 2009

Balanceo de Líneas de Ensamble de Modelo Mixto Mediante Encadenamiento de Heurísticas

Halley Jarumy Ferrer Soriano, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2009

Asesor de la tesis: Dr. Santiago E. Conant Pablos

El trabajo de investigación que se presenta en esta tesis de la Maestría en Ciencias con especialidad en Sistemas Inteligentes se enfoca a la solución del problema de balanceo de líneas de ensamble de modelos mixtos. Dicho problema proviene de los requerimientos de los mercados actuales que se encuentran cambiando constantemente y como resultado los sistemas de producción deben ser lo suficientemente flexibles para soportar la variabilidad de la demanda, lo que da como resultado una misma línea de ensamble para producir diferentes productos y de esa manera nos enfrentamos al problema de balancear estas líneas. En esta tesis se desarrollaron tres algoritmos basados en el encadenamiento de heurísticas, como son Recocido Simulado, Min-Conflicts y Algoritmos evolutivos. Igualmente se implementó un generador de problemas que fue de gran ayuda para realizar experimentos con los algoritmos implementados. Después de realizar la comparación entre ellos se eligió el modelo de solución, el cual contempla tres etapas; durante la primera etapa se hace uso de heurísticas que forman parte de los métodos clásicos de balanceo de líneas; la siguiente etapa incluye un algoritmo memético, el cual utiliza la técnica recocido simulado para llevar a cabo la mutación de los individuos, durante esta etapa se trata de maximizar el nivel de balanceo de la línea; y finalmente, la tercera etapa contiene una variación del algoritmo Min-Conflicts que propicia la obtención de una solución que se adapta mejor a las preferencias del usuario sin descuidar la parte de la eficiencia de la línea. Como resultado de este encadenamiento de heurísticas, se obtuvieron soluciones competentes, muy cercanas al óptimo.

Índice general

Índice de cuadros

Índice de figuras

Índice de algoritmos

Capítulo 1

Introducción

La optimización de los procesos de producción es una prioridad en los actuales planes de negocio de las empresas. Esta optimización se logra a partir de la reducción sistemática de los desperdicios; Taiichi Ohno, pionero del Sistema de Producción Toyota, identificó los principales orígenes del desperdicio que él denominó "Los Siete Desperdicios Mortales". Dentro de estos desperdicios se encuentra la demora o espera que indica una parada entre dos operaciones sucesivas. Cuando tratamos de balancear una línea de ensamble esperamos que todas las estaciones contengan la misma cantidad de trabajo, ya que así el producto se mueve a través de ellas a un mismo tiempo de ciclo. Si todas las estaciones terminan al mismo tiempo, no habrá espera, y si por el contrario una estación está sobrecargada, el resto de las estaciones tendrán que esperar para poder continuar con el siguiente producto.

Una línea de ensamble es un sistema de fabricación donde el producto es llevado a lo largo de las estaciones en las que se le agregan partes de tal manera que al final de línea se obtenga un producto completo. Los productos son transportados a través de las estaciones mediante algún sistema de manejo de materiales como las bandas transportadoras. En cada estación de trabajo se realiza un conjunto de tareas que cumplen un proceso predefinido de ensamblado. El Problema de Balanceo de Líneas de Ensamble (Assembly Line Balancing Problem - ALBP) consiste en distribuir cada tarea u operación de trabajo entre todas las estaciones disponibles de tal manera que el esfuerzo de trabajo sea asignado a una estación en la misma medida que al resto de las estaciones, tanto como sea posible. Las tareas que serán asignadas a las estaciones tienen restricciones de precedencia, restricciones ergonómicas o restricciones de incompatibilidad de tareas lo que dificulta obtener una asignación equitativa de la carga de trabajo entre ellas.

Cuando se tiene una misma línea de ensamble para producir diferentes productos con una secuencia arbitraria de fabricación de los diferentes artículos, entonces esta línea es llamada *Línea de Ensamble de Modelo Mixto* (Mixed-model Assembly Line [21]). Esta forma de producción ha reemplazado a la producción en masa tradicional para dar respuesta a las expectativas de los clientes actuales. Paralelamente al surgimiento de las Líneas de Ensamble de Modelo Mixto surge el problema de balancear dichas

líneas conocido como *Problema de Balanceo de Líneas de Modelo Mixto* (Mixed-Model Assembly Line Balancing Problem - MALBP). Este problema es conocido como un problema de optimización NP-completo [6].

El problema de balanceo de líneas de ensamble ha llamado la atención de varios investigadores a lo largo de varias décadas debido a que es un problema de gran importancia para las industrias de manufactura tales como la industria Automotriz, la industria Electrónica o cualquier otra con sistemas con líneas de flujo. La formulación matemática para el problema de balanceo en líneas de ensamble de un solo producto fue establecido por Salveson en 1955 [27], según comenta Vilarinho [15], y en realidad el primero en introducir este problema fue Bryton en su tesis en 1954 [3]. Desde entonces, se han desarrollado heurísticas y otros algoritmos basados en técnicas de Inteligencia Computacional para tratar de dar solución a los diferentes tipos de problemas de balanceo de líneas de ensamble. La importancia de balancear estas líneas recae en el hecho de que al llevar a cabo este balanceo de líneas de ensamble se ha comprobado que mejora diferentes medidas de desempeño de los sistemas de producción tales como, minimización del número de estaciones de trabajo para un tiempo de ciclo dado, minimización del tiempo de ciclo para un número dado de estaciones, suavización de la sobrecarga de trabajo y maximización de la eficiencia de las líneas de modelo mixto [12].

Por otro lado, el balanceo de líneas usualmente se realiza de manera manual, lo cual toma una gran cantidad de tiempo y recursos, ya que este problema no es sencillo de resolver y se va complicando aún mas con las características particulares de cada tipo de problema; por ejemplo, Schmitz et al. [9] menciona que para un rebalanceo tardan alrededor de 12 semanas considerando que cuentan con 14 zonas, en promedio 20 estaciones, esto en una planta de ensamble de Hermosillo, Sonora.

En esta investigación se desea obtener un algoritmo basado en técnicas de Inteligencia Computacional que resuelva el balanceo de manera eficiente, en un periodo de tiempo razonable. Estas técnicas de inteligencia computacional de acuerdo a la Sociedad de Inteligencia Computacional de la IEEE, definen su campo de interés como Redes Neuronales (NN), Lógica Difusa (FL), Algoritmos Evolutivos (EA) y también el uso de otros enfoques con menor difusión. Para resolver este problema de balanceo de líneas se utilizan Algoritmos Genéticos y la técnica Min-Conflict, con lo que determinamos que es factible la aplicación de dichos algoritmos para el problema planteado de balanceo de líneas, sin embargo, los mejores resultados obtenidos en todo caso dependerá de las preferencias del usuario o los requerimientos del problema.

Antes de llevar a cabo el balanceo de línea, existen dos requisitos que se deben cumplir:

- Estandarización del trabajo; reduce variaciones del proceso, facilita la formación de nuevos operarios y además reduce las posibilidades de accidentes y lesiones.
- Análisis del *Takt Time*; establece el ritmo al que se deben hacer distintos pro-

ductos de un proceso para satisfacer la demanda del cliente.

Es importante tomar en cuenta que el balanceo perfecto raramente es encontrado, no solamente porque no se encuentre una combinación de tareas que cumpla con esta característica sino porque no todas las tareas están automatizadas, lo cual resulta en una variabilidad de los tiempos de ejecución de las operaciones. Otro factor que afecta la calidad de balanceo obtenida, resulta de las restricciones consideradas durante la solución del problema. Dada esta naturaleza del problema, se considera adecuada la aplicación de técnicas de inteligencia computacional, como los algoritmos genéticos que permiten optimizar el nivel de balanceo de una solución y la técnica de satisfacción de restricciones Min-Conflicts que encuentra soluciones con menos conflictos en restricciones.

1.1. Definición del Problema

La principal clasificación de los ALBPs es: i) Problema de balanceo de líneas de ensamble de un solo producto (Single Assembly Line Balancing Problem - SALBP), ii) Problema de balanceo de líneas de ensamble multimodelo (Multi-Model Line Balancing Problem) y iii) Problema de balanceo de líneas de ensamble de modelo mixto (Mixed-Model Assembly Line Balancing Problem MALBP). El primer tipo de problema ha sido el más estudiado de todos, sin embargo no refleja la complejidad de los sistemas de ensamble actuales. El segundo tipo de problema se refiere a un sistema de producción por lotes. El presente trabajo de investigación será enfocado al tercer tipo de problema que como se había mencionado en párrafos anteriores, consiste en balancear una línea que produce más de un producto sin tomar en cuenta cambios de herramental para producir los diferentes productos [28]. Y aunque ya se han propuesto varios métodos, que se verán a mayor detalle en el capítulo de antecedentes, para resolver este problema aún existen áreas de oportunidad en esos algoritmos, entre los que se encuentran la brecha existente de los resultados obtenidos de esos métodos y las soluciones óptimas, y restricciones adicionales a las de precedencia que usualmente se encuentran en las líneas de ensamble actuales, de tal manera que este trabajo se concentrará en estos dos aspectos.

Una instancia de este tipo de problemas consiste en proporcionar el conjunto de tareas que se desea distribuir entre las estaciones, cada tarea con sus respectivas restricciones y tiempo de procesamiento para cada producto, también proporcionar el número máximo de estaciones que se utilizarán y finalmente delimitar la proporción de la demanda de todos los productos. Dentro de las restricciones que se tomarán en cuenta están, las restricciones de precedencia que establecen conjuntos de tareas que deben concluirse antes de ejecutar una tarea en particular; restricciones de incompatibilidad de tareas, que establecen pares de tareas que no pueden ser desarrolladas en

una misma estación; estas restricciones se tomaron como restricciones duras, esto es, en ningún caso pueden ser violadas. Adicionalmente también se consideran restricciones ergonómicas para evitar condiciones inseguras para los operadores, y restricciones propias de las estaciones relacionadas con el equipo instalado o el espacio disponible en cada estación. Estas restricciones son consideradas blandas, en el sentido de que establecen preferencias que se pueden cumplir o no durante el proceso de solución del problema.

Los objetivos que debe cumplir la solución al MALBP son: minimizar el número de estaciones utilizadas, maximizar la eficiencia de la línea de ensamble. Otro objetivo deseable es la maximización del *throughput*, es decir, de la velocidad de producción que está dado por el tiempo de ciclo de las estaciones. Para nuestro caso particular se utiliza una función objetivo que busca minimizar las restricciones suaves violadas y maximizar la eficiencia de la línea.

Adicionalmente existen otras variables relacionadas con los operadores que pueden afectar las decisiones de distribución de las tareas y para el presente trabajo de investigación se asumirá que no tienen ningún efecto; entonces, suponemos que los operadores que estarán trabajando en las estaciones ya están entrenados, por lo cual los efectos de aprendizaje o mejoras sucesivas de las habilidades en el proceso productivo no se tomarán en cuenta [2]. Tampoco se consideran líneas de alimentación de materiales, ni una distribución específica del área de ensamble, por lo que se supone que las estaciones están distribuídas de manera serial, a lo largo de una cinta transportadora [1].

La Figura 1.1, tomada de Scholl y Becker [1], ilustra los diferentes tipos de líneas de ensamble; el inciso *a)* muestra una línea de un solo producto, en el inciso *b)* la línea de ensamble de modelo mixto, finalmente en el inciso *c)* se muestra una línea multimodelo.

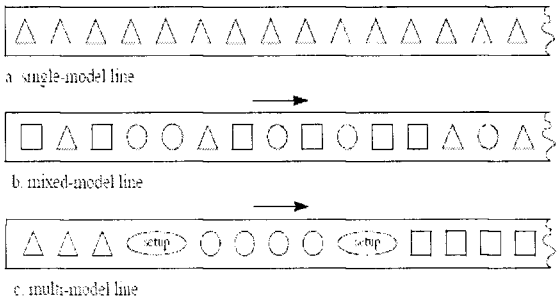

Figura 1.1: Tipos de líneas de ensamble

1.2. Motivación

El balanceo de línea ha demostrado ser de mucha utilidad en diversos tipos de industrias, por lo que desde su formulación ha sido muy estudiado. Se han desarrollado varios métodos para la solución de este problema. Algunos se han enfocado en la solución del problema de balanceo de un solo producto y otros a la solución del problema con varios productos. Sin embargo, aún sigue existiendo una brecha entre los métodos desarrollados y los problemas que existen en las industrias. Típicamente los problemas se simplifican y algunos aspectos importantes de los problemas reales son ignorados.

Algunos métodos asumen que el balanceo se hará sobre una línea nueva y no sobre una ya existente, por lo cual comienzan a crear y crear estaciones, cuando en realidad éstas ya existen; además, el crear una nueva estación implica una fuerte inversión por lo que es una decisión que se debe evaluar a profundidad.

Otro aspecto a considerar es la necesidad de ubicar tareas en ciertas estaciones, esto se presenta normalmente cuando se tiene instalado un equipo grande y pesado que el moverlo de estación resulta costoso. Esto significa que las operaciones que requieran dicho equipo deben ser asignadas a la estación que cuenta con él.

Una dificultad adicional que usualmente no se considera dentro de los métodos desarrollados se refiere a las restricciones ergonómicas, esta situación se presenta debido a las características del producto y del proceso. Por ejemplo, algunas veces es necesario mover objetos de un lugar a otro, que combinados la repetitividad y el peso de los objetos podrían representar un riesgo ergonómico para el operador que desarrolla la operación.

1.3. Objetivos

El principal objetivo de la tesis consiste en desarrollar un algoritmo que sea capaz de balancear líneas de ensamble de modelos mixtos con tiempos determinísticos obteniendo resultados comparables a los obtenidos por algoritmos previamente desarrollados en un tiempo razonable y además considerando restricciones adicionales, todo esto mediante el uso de técnicas de Inteligencia Computacional.

1.3.1. Objetivos Específicos

- Implementar los modelos clásicos utilizados para realizar balanceo de líneas, Hegelson y Birnie [11] y Kilbridge y Wester [13].
- Determinar las técnicas que nos aseguren obtener mejores resultados que los modelos en la literatura como son, Hegelson y Birnie [11], Kilbridge y Wester [13], y Moodie y Young [20].

- Estudiar un modelo propuesto que resuelva un tipo de problema similar al que se desea resolver.
- Implementar un generador de problemas para realizar pruebas con el modelo que fue desarrollado.
- Encontrar una buena forma para manejar las restricciones del problema de balanceo.
- Encontrar la función objetivo que mejor refleje lo que deseamos como resultado del algoritmo.

1.4. Hipótesis

La hipótesis principal de la presente tesis, es que utilizando técnicas de Inteligencia Computacional es posible obtener buenos resultados en el balanceo de líneas de ensamble mixtas.

Las preguntas que nos servirán como pauta para la investigación son:

- ¿Cuál técnica de Inteligencia Computacional debido a sus características particulares se adapta a las necesidades del problema de balanceo de líneas?
- ¿Qué función objetivo representa la salida que deseamos obtener?
- ¿De qué manera conviene realizar la selección de tareas que serán distribuídas en las estaciones?
- ¿Es posible implementar un algoritmo que funcione satisfactoriamente tanto en problemas de balanceo sencillos como en problemas mas complejos?

1.4.1. Justificación

El balanceo de líneas de ensamble es un problema de gran importancia y aunque en la actualidad existen diversas formas para resolver este problema, muchos de estos métodos no se ajustan a las características reales de los problemas, ya que simplifican el problema original para poder darle solución. Por otro lado, cuando el balanceo se realiza de forma manual consume mucho tiempo y recursos, por lo cual es necesario obtener un algoritmo que sea capaz de encontrar soluciones en un tiempo más corto.

Tomando en consideración lo anterior se puede justificar esta investigación y se fortalece la idea de que la hipótesis propuesta es correcta.

1.5. Contribución

En esta investigación se contribuye a la mayor comprensión del problema de balanceo de líneas de ensamble de modelos mixtos, particularmente en los siguientes aspectos:

- La existencia de un algoritmo y una función objetivo simples para obtener buenas soluciones comparables con las obtenidas por otros métodos.
- La efectividad de separar el manejo de las restricciones de precedencia del resto de las restricciones consideradas.
- Las ventajas de maximizar la eficiencia mediante un algoritmo memético y posteriormente tratar las restricciones suaves sobre un solo individuo utilizando Min-Conflicts.
- Variaciones de los algoritmos de Recocido Simulado, Min-Conflicts y Algoritmo Genético para adaptarlos a las necesidades del problema de estudio.

1.6. Organización de la Tesis

A continuación se presenta una descripción general de la organización de la presente tesis:

En el Capítulo 2 se presentan los antecedentes vinculados con el problema de balanceo de líneas de ensamble de modelo mixto. El Capítulo 3 presenta el modelo de solución propuesto, en él se expone la descripción a detalle de los algoritmos implementados. En el Capítulo 4 se presentan los primeros pasos del desarrollo del modelo de solución, algunos experimentos realizados y los resultados obtenidos; también se describe un generador de problemas desarrollado para crear las instancias que fueron utilizadas para llevar a cabo los experimentos tanto preliminares como finales. El Capítulo 5 presenta los experimentos y resultados obtenidos después del proceso de refinamiento del modelo de solución. Finalmente, en el Capítulo 6 se presentan las conclusiones y se discuten las sugerencias para trabajos futuros.

1.7. Resumen

En este capítulo se presentó una breve introducción al trabajo de tesis, también se describió el problema que se tratará así como los objetivos y las preguntas de investigación que sirvieron como guía durante el desarrollo del presente trabajo. Ahora que se conoce el problema que se desea resolver, en el siguiente capítulo se exponen los

antecedentes de dicho problema los cuales sirvieron como marco de referencia para esta investigación.

Capítulo 2

Antecedentes

En este capítulo se presenta el marco teórico que sustenta este trabajo de tesis. Los temas presentados en este capítulo son necesarios para la comprensión de la investigación y sus implicaciones. Primero se describe la clasificación de los problemas de balanceo de líneas de ensamble, así como los métodos principales para tratar con problemas de este tipo. Finalmente se describen de forma breve los trabajos relacionados con la presente investigación.

2.1. Introducción

Las líneas de ensamble son sistemas de producción basados en líneas de flujo y aunque el balanceo de estas líneas ha atraído a muchos investigadores, los modelos desarrollados se han enfocado principalmente en aquellos en los que se fabrica un solo producto y solo unos cuantos se enfocan en problemas más reales [22]. Algunos de estos modelos van a ser descritos en esta sección. Igualmente algunos conceptos y antecedentes del ALBP serán definidos.

Fabricar un producto en una línea de ensamble requiere que la carga total de trabajo esté dividida en un conjunto de elementos llamados tareas. Cada tarea se realiza en un tiempo dado, y requiere cierto tipo de equipo y maquinaria además de ciertas habilidades de los operarios. Estas tareas deben ser repartidas entre las estaciones de la línea de ensamble, entonces el problema consiste en encontrar una línea balanceada factible, esto es una asignación de cada tarea a una estación tal que cumpla con todas las restricciones de precedencia y de preferencia cumpla también con restricciones adicionales como las ergonómicas. El conjunto de tareas asignadas a una estación constituye la carga de trabajo de dicha estación y el tiempo acumulado de las tareas dentro de una estación de trabajo es el tiempo de la estación. Este tiempo no debe exceder el tiempo de ciclo; en caso de que el tiempo de una estación sea menor al tiempo de ciclo, entonces la estación tiene un tiempo muerto durante el cual no se está desarrollando ninguna operación.

El balance de línea debe realizarse durante la preparación inicial de la misma,

cuando una línea se balancea para modificar su tasa de producción por hora, o cuando se introducen cambios en el producto o el proceso. Y debido a que la introducción de nuevos productos o bien la instalación de una línea de ensamble nueva es una actividad que conlleva una gran inversión de capital es imprescindible que las líneas estén diseñadas y balanceadas correctamente de tal manera que trabajen tan eficientemente como sea posible.

2.2. Ejemplo de balanceo de una Instancia del ALBP

En el problema base de balanceo de líneas se consideran restricciones de precedencia y dado un número definido de estaciones debemos encontrar un balanceo óptimo que resulte en el menor tiempo de ciclo requerido. En la siguiente tabla 2.1 se muestra un ejemplo para un solo modelo, en la primera columna se encuentra el número de tarea, la siguiente columna muestra el tiempo que tarda en realizarse cada tarea y la última columna muestra las restricciones de precedencia de estas tareas.

Cuadro 2.1: Ejemplo de una instancia del ALBP

n	Tiempo ejecución *(Ti)*	Precedencia
1	5	-
2	3	-
3	6	1
4	8	1,2
5	10	3,4
6	7	4
7	1	5,6
8	5	7
9	3	7
Total	48	-

El diagrama de precedencia se muestra en la figura 2.1. Por ejemplo, se observa en la tabla la tarea 8 que tiene precedente a la tarea 7, esto se observa en el grafo también en el que ambas tareas están conectadas por un arco.

Ahora bien, para resolverlo, supongamos que tenemos 3 estaciones disponibles (k=3) en las que debemos repartir estas tareas. Físicamente, el área de trabajo existente con las 3 estaciones estaría como se muestra en la figura 2.2. Entonces el tiempo de ciclo óptimo TO para este problema debe ser:

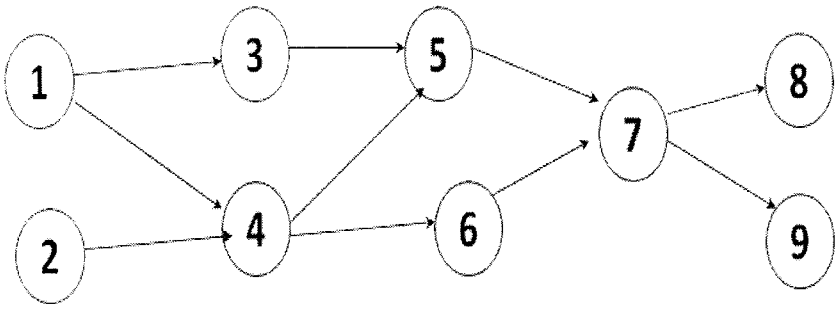

Figura 2.1: Diagrama de precedencia del ejemplo de una instancia del ALBP

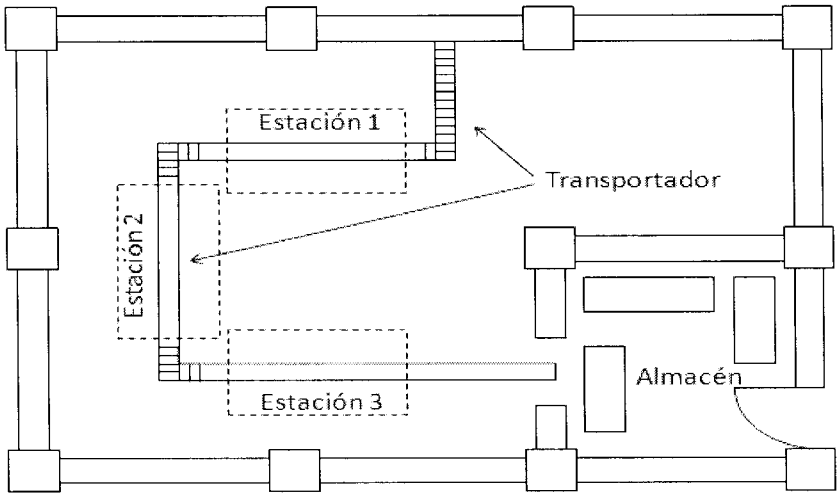

Figura 2.2: Distribución de estaciones de trabajo.

$$TO = \frac{\sum_{i=1}^{n} Ti}{k}$$

$$TO = \frac{48}{3} = 16$$

Debido a que este ejemplo es bastante sencillo, por simple observación de la tabla 2.1 es posible agrupar las tareas de tal manera que el tiempo de ciclo de cada estación sume exactamente 16 unidades de tiempo. Si se agrupan estas tareas como se muestra en la figura 2.3, todas las estaciones tendrían un tiempo de ciclo de 16 y por lo tanto se obtiene un balanceo perfecto.

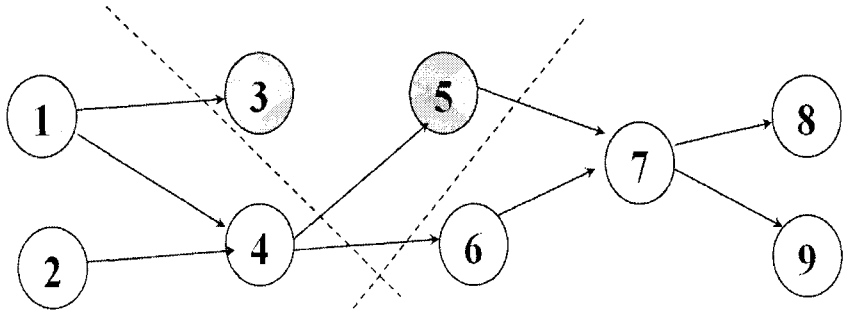

Figura 2.3: Solución al ejemplo de una instancia del ALBP

En este caso cuando aplicamos la función objetivo 3.2:

$$B = (eficiencia * 0.8) + (vrs * 0.2) = (1 * 0.8) + (1 * 0.2) = 1$$

de donde,

$$eficiencia = 1 - \frac{\sum_{k=1}^{S} abs\langle to - \sum_{m=1}^{M} q_m S_{km} \rangle}{to * S}$$

$$eficiencia = 1 - \frac{\sum_{k=1}^{S} abs\langle 16 - 16 \rangle}{16 * 3}$$

$$eficiencia = 1 - \frac{0}{48} = 1$$

La variable *vrs* toma el valor de 1 en este caso, al no tener ningún conflicto debido a restricciones ergonómicas.

Finalmente la codificación del cromosoma se muestra en el cuadro 2.2, este cromosoma es de tamaño n, en este caso es de tamaño 9. La distribución quedó de la siguiente forma: las tareas 1, 2 y 4 se realizan en la estación 1; las tareas 3 y 5 se realizan en la estación 2; y las tareas 6 a 9 se realizan en la estación 3.

Cuadro 2.2: Cromosoma de distribución óptima de tareas del ejemplo de instancia del ALBP

1	1	2	1	2	3	3	3	3

2.3. Complejidad de las Instancias del Problema de Balanceo de Líneas de Ensamble

Los algoritmos usualmente resuelven una gran cantidad de instancias en periodos de tiempo cortos, mientras que otros son computacionalmente intratables en términos del tiempo disponible. Un algoritmo en particular pudiera resolver una instancia rápidamente, o no, dependiendo de la complejidad de dicha instancia. Para cuantificar estos aspectos de complejidad se presentan a continuación algunas características e indicadores usualmente utilizados. En general se espera que aquellas instancias que cumplen alguna o ambas de las siguientes propiedades se resuelvan de manera favorable respecto a los requerimientos de tiempo computacional.

- El número de soluciones óptimas es grande y dichas soluciones se encuentran esparcidas en el espacio de solución. En este caso, la probabilidad de encontrar la solución óptima se incrementa con el número existente de soluciones óptimas.
- El valor óptimo de la función objetivo es conocido. Cuando conocemos el valor óptimo de la función objetivo el proceso de búsqueda puede detenerse, a diferencia de no conocer este valor, en el cual deberá transcurrir un mayor tiempo para probar que una solución encontrada es la óptima.

Adicionalmente, las siguientes características tienen una fuerte influencia en la complejidad de una instancia:

Número n de tareas: la complejidad crece exponencialmente con el incremento del número de tareas. Si ignoramos las restricciones de precedencia, tendríamos en total $n!$ secuencias de tareas posibles.

Tiempo de las tareas y tiempo de ciclo: cuando los tiempos de las tareas son más cortos que el tiempo de ciclo existen más probabilidades de que existan más soluciones óptimas, lo cual representa una gran ventaja.

Restricciones de precedencia: no solo la cantidad de restricciones sino la estructura de las restricciones de precedencia tienen influencia en la complejidad de las instancias. Para medir esto se tienen dos indicadores: el grado de convergencia

y el grado de divergencia. Un grafo dirigido tiene un grado de divergencia igual a 1 cuando cada nodo esta conectado por un solo arco. partiendo desde la raíz. Entonces se dice que es estrictamente divergente. Cuando el grado de divergencia es menor pero cercano a 1 se le llama casi-divergente. Entonces las instancias con un alto grado de divergencia son menos complejos que aquellos con un grado de divergencia menor. El grado de convergencia es análogo al grado de divergencia, la diferencia recae en que se analiza el grafo de reversa, es decir, no se parte de la raíz sino de la meta. De la misma manera entre más grande el grado de convergencia menos complejo es la instancia que se intenta resolver.

Número m de estaciones: cuando existen menos estaciones disponibles, existen más combinaciones de tareas que deben ser examinadas para cada estación. Como consecuencia, la complejidad se incrementa cuando m decrece.

West ratio: la medida *West ratio* $WR = \frac{n}{m}$ es el número promedio de tareas por estación. Esto significa que para valores pequeños de WR las instancias tienden a ser más complejas que aquellas con valores más grandes [7].

Radio de variabilidad de tiempo (Rv): esta medida es independiente del tiempo de ciclo ya que $Rv = \frac{t_{max}}{t_{min}}$, donde el numerador es el tiempo mayor encontrado en el conjunto de los tiempos de las tareas y el denominador es el tiempo menor dentro del conjunto de los tiempos de las tareas. Se espera que la complejidad de la instancia crezca mientras decrece el valor de Rv.

2.4. Artículos Relacionados con el Problema de Balanceo de Líneas de Ensamble (ALBP)

El problema de Balanceo de Líneas de Ensamble ha sido investigado durante varias décadas, aunque la formulación matemática para el problema de balanceo en líneas de ensamble de un solo modelo fue establecido por Salveson en 1955 [27], según comenta Vilarinho [15], en realidad el primero en introducir este problema fue Bryton en su tesis en 1954 [3].

Algunos de los métodos más importantes que se desarrollaron en las primeras décadas de investigación incluyen la técnica de ranqueo posicional de Hegelson y Birnie [11], la heurística de Kilbridge y Wester [13], la heurística de Moodie y Young [20] que serán descritas a mayor detalle en las siguientes secciones. Estas técnicas tienen en común el hecho de que son utilizadas para la solución del problema simple de balanceo de líneas de ensamble. A continuación se presentan algunos de los métodos importantes que se desarrollaron durante los primeros años de investigación.

2.4.1. Métodos Clásicos de Solución del ALBP

Los métodos clásicos que se desarrollaron durante los primeros años de investigación en este tema fueron métodos heurísticos. La palabra heurístico proviene de la palabra griega *Heuriskein* que significa descubrir. Los heurísticos son un conjunto de reglas que tratan de descubrir una o más soluciones específicas de un problema determinado. Estas reglas están basadas en razonamientos deductivos de personas, debido a su intuición, conocimiento y experiencia. Por lo general los heurísticos se construyen para darle apoyo al algoritmo en los problemas que tienen dimensiones grandes. Enseguida se presentan dichos métodos.

Hegelson y Birnie

Hegelson y Birnie en 1961 [11] desarrollaron la "Técnica de Ranqueo de Pesos Posicionales". Este método consiste en estimar el peso posicional de cada tarea como la suma de su tiempo más los de aquellas que la siguen. La principal desventaja de este método es que únicamente toma en cuenta las restricciones de precedencia, sin embargo es importante mencionarlo ya que ha sido utilizado en varias investigaciones para realizar la selección de una tarea a asignar ([16], [17], [15]). El algoritmo 1 muestra los pasos a seguir para utilizar el método de Hegelson y Birnie.

Algoritmo 1 Algoritmo de Hegelson y Birnie

1: Calcular los pesos posicionales de cada tarea como la suma de su tiempo más el tiempo de las tareas sucesoras.
2: Ordenar las tareas de mayor a menor peso posicional.
3: Las tareas se asignan a las estaciones de acuerdo al peso posicional, cuidando no rebasar el tiempo de ciclo y violar las precedencias. Cuando una estación ya no tiene suficiente tiempo para la siguiente operación entonces se abre una nueva estación y se asigna en ésta la operación actual.

Kilbridge y Wester

Kilbridge y Wester en 1962 [13] desarrollaron un método heurístico de balanceo de línea el cual se describe en el algoritmo 2.

Moodie y Young

Moodie y Young en 1965 [20] desarrollaron una heurística que consiste en repartir las tareas en las estaciones de trabajo eligiendo primero las tareas que tenga el tiempo de procesamiento más largo. Esta heurística al igual que las otras toma en cuenta las

Algoritmo 2 Algoritmo de Kilbridge y Wester

1: A partir de un diagrama de precedencia, con este diagrama se forman columnas donde la primera contiene todas las tareas sin ninguna tarea precedente y las columnas siguientes contendrán tareas que tenían como requerimiento alguna actividad de la columna previa.
2: Determinar el tiempo de ciclo de las estaciones; este tiempo de ciclo debe ser un entero y mayor que el tiempo de duración de una tarea y menor que la suma de todos los tiempos de las tareas.
3: Realizar la asignación de las tareas, dentro de cada columna en orden asignando primero el elemento con mayor duración, a menos que no haya suficiente tiempo disponible en la estación para éste, pasarse a las tareas con menor duración. Una vez que ya se hayan asignado las tareas de una columna, pasarse a la siguiente en orden ascendente.

restricciones de precedencia de modo que una tarea no puede ser asignada a menos que todas sus predecesoras ya hayan sido asignadas.

Aunque los métodos descritos previamente se habían enfocado en resolver el problema de balanceo de líneas de ensamble de un solo producto resultaron ser además de simples realmente efectivos; posteriormente se desarrollaron otros modelos que consideraban tanto el problema de balanceo de líneas de ensamble de modelos-mezclados como el problema de balanceo de líneas multi-modelo. Además se han estudiado otras variantes de dichos problemas como aquellos que consideran las estaciones paralelas, la variabilidad de los tiempos de tareas, la distribución de las líneas y los métodos alternos de procesamiento. Uno de los análisis más recientes de los diferentes algoritmos existentes es el realizado por Becker y Scholl en 2006 [1].

2.4.2. Métodos Basados en Técnicas de Inteligencia Computacional de Solución del ALBP

Trabajos de investigación más recientes comenzaron a incluir aspectos relevantes como el uso de estaciones paralelas [15], producción de varios modelos [15],[6], [12],[18],[5],[30], procesos alternos, el uso de tiempos de procesamiento estocásticos [18], distribución de líneas diferente a la serial como la forma U o las líneas alimentadoras [14] entre otros. Uno de los trabajos actuales que contempla un resúmen bastante completo de estos modelos es el publicado por Becker y Scholl [1]. Debido a que el problema que se trata de resolver se refiere al balanceo de líneas de ensamble de modelos mixto veremos la descripción de algunos modelos desarrollados para resolverlo.

McMullen y Frazier [18] propusieron una heurística para la solución del problema de balanceo de líneas de modelos mixtos con el uso de tiempos estocásticos y estaciones paralelas. En este modelo, la duración de cada tarea es calculada en base a la mezcla

de la demanda de cada producto y su duración correspondiente, luego se genera una lista y mediante procedimientos de selección y aceptación las tareas son asignadas a las estaciones.

Bukchin et al [4] en su investigación denominada "Diseño de una Línea de Ensamble de Modelo Mixto en un Ambiente Make-To-Order (MTO)", que significa hacer de acuerdo a instrucciones especiales, presentaron una heurística que minimiza el número de estaciones para un tiempo de ciclo predeterminado. Éste consiste en tres etapas; balanceo de un diagrama de precedencias combinado, balanceo de cada modelo de forma separada y una búsqueda en la vecindad basada en un procedimiento de mejora.

Vilarinho y Simaria [15] desarrollaron un método heurístico de dos etapas también para el problema de balanceo de líneas de ensamble de modelos mixtos con estaciones de trabajo paralelas. Presentaron un modelo matemático en el cual el objetivo principal era minimizar el número de estaciones y como objetivo secundario era balancear la carga de trabajo. Utilizaron un procedimiento de dos etapas mediante un enfoque de recocido simulado. Al principio el algoritmo busca una solución sub-óptima y después se concentra en el objetivo secundario.

McMullen y Tarasewich [19] presentaron un enfoque basado en una técnica de colonia de hormigas, donde además incluyeron los factores de duración de tareas estocástico, estaciones de trabajo paralelas y modelos mixtos. Esta metodología fue inspirada en el comportamiento social de los insectos para distribuir las tareas entre los trabajadores de manera que su medida de desempeño es optimizada. Los resultados de este enfoque fueron comparados con varias heurísticas como las de recocido simulado. Esta comparación mostró que el enfoque utilizando colonia de hormigas es competitivo con los demás métodos heurísticos.

Bukchin y Rabinowitch [5] resolvieron este problema mediante una solución basada en un enfoque ramificación y poda. Este modelo relaja la restricción de asignar tareas comunes entre los modelos a la misma estación y minimizar el costo de generar una nueva y el de duplicación de la tarea. Aplicando su técnica propuesta, se encuentra rápidamente una solución con el valor objetivo del costo total. Después comienza la búsqueda de otras soluciones mediante un sondeo comenzando por las ramas vecinas. Los nodos del árbol representan soluciones parciales.

Otros modelos desarrollados han utilizado la técnica de optimización de Colonia de Hormigas [17], Algoritmos Genéticos [26],[24],[16] y Búsqueda Tabú [10] la cual es una búsqueda estocástica que utiliza una lista para registrar la historia y buscar en otra dirección y con ello evitar quedarse atrapado en un mínimo local.

Debido a que el trabajo de tesis se concentrará en el MALBP, se estudiaron principalmente los modelos desarrollados en esta clasificación. Finalmente se encontró un modelo muy parecido a lo que se desea lograr; "A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II"desarrollado por Vilarinho P. M. y Simaria A.S. [16]. Este algoritmo contiene muchas de las características que

estamos considerando, de manera que fue tomado como base para comenzar a desarrollar nuestro método. En el siguiente capítulo en el modelo de solución se describe el algoritmo para obtener la población inicial del algoritmo evolutivo que fue programado a partir del de Vilarinho y Simaria.

2.5. Técnicas Utilizadas para la Solución

En esta sección se presentarán algunas técnicas de Inteligencia Computacional que se pretenden utilizar para resolver el problema de balanceo de líneas de ensamble de modelos mixtos. La solución de este problema no solamente debe cumplir con las restricciones impuestas en cada instancia sino que también debe estar muy cercana a una solución óptima, por lo cual se requiere una técnica o una combinación de técnicas que sean capaces de manejar estos dos requerimientos.

2.5.1. Heurística Min-Conflicts

Es un algoritmo de búsqueda local que se mueve de una asignación infactible hacia una factible. El criterio para seleccionar un nuevo valor para una variable es elegir aquel valor que resulte en un número menor de conflictos con otras variables. La etapa inicial pudiera ser elegida aleatoriamente o mediante otro proceso de asignación que elija valores que tengan un mínimo de conflictos para cada variable.

La función de conflictos cuenta el número de restricciones violadas por un valor en particular. A continuación, en el algoritmo 3 se presenta el algoritmo base de Min-Conflicts tomado del libo AIMA [25]. Es importante mencionar que en el algoritmo Min-conflicts el tiempo de procesamiento es independiente del tamaño del problema.

Otro punto importante acerca de esta técnica y que no debemos pasar por alto es el hecho de que no es capaz de dejar un mínimo local, por lo cual una vez que alcanza este valor ya no le será posible mejorar más. Para nuestro caso de estudio este algoritmo nos ayudara para el problema de satisfacción de restricciones. Para resolver el otro problema de optimización, se proponen dos técnicas Algoritmos Evolutivos y Recocido Simulado.

2.5.2. Algoritmos Evolutivos

El gran campo de aplicación de los Algoritmos Evolutivos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Evolutivos.

Para desarrollar los algoritmos que fueron base para obtener el modelo de solución se utilizaron un algoritmo genético y un algoritmo memético que se encuentran dentro

Algoritmo 3 Algoritmo Base de Min-Conflicts

1: **función** MIN-CONFLICTS $\langle csp, max\_interaciones \rangle$ **regresa** una solución o falla
entradas: *csp*, problema de satisfacción de restricciones
max_iteraciones, numero de repeticiones permitidas
actual $\leftarrow$ una asignación inicial completa del *csp*

1. **Para** $i = 1$ hasta *max_iteraciones* **hacer**

Si *actual* es una solución para *csp* **entonces regresa** *actual*

var $\leftarrow$ *a* elegida aleatoriamente, variable conflictiva de VARIABLES(*csp*)

valor $\leftarrow$ *el* valor *v* para *var* que minimize CONFLICTS(*var,v, actual, csp*)

asignar *var* = *valor* en *actual*

regresar *falla*

de los algoritmos evolutivos, con algunas diferencias entre sí que se describirán a mayor detalle en las siguientes secciones.

Conceptos básicos

Individuo o cromosoma: Un individuo determina una potencial solución del problema que se pretende resolver mediante el algoritmo evolutivo.

Población: Conjunto de individuos con los que se trabaja en el algoritmo. En un algoritmo evolutivo los individuos que constituyen la población van cambiando pero generalmente el tamaño de la misma permanece constante.

Función de evaluación: Se trata de una función evaluadora de la calidad de un individuo como solución a nuestro problema. Permite la ordenación de los individuos de la población en cuanto a bondad de los mismos.

Cruce: Es una de las operaciones fundamentales que intervienen en todo algoritmo genético. Como norma general se aplica después de un proceso de selección de dos individuos y consiste en una combinación de los mismos para obtener como resultado otros dos nuevos individuos. Existen dos técnicas principales, intercambio a partir de un solo punto de cruce y a partir de dos puntos de cruce. En ambas técnicas se obtienen dos descendientes fruto del cruce.

Mutación: Constituye otra operación fundamental, en el caso del algoritmo genético se selecciona un individuo, el cual sufre una pequeña modificación aleatoria en su codificación obteniéndose otro individuo nuevo. Por otro lado, en el algoritmo memético para nuestro caso particular la mutación se realiza mediante otro proceso de búsqueda local, el recocido simulado.

Gen: Partícula de los cromosomas que producen la aparición de caracteres hereditarios.

Generación: Iteración de la medida de aptitud y la creación de una nueva población por medio de operaciones genéticas.

Para diseñar un algoritmo evolutivo se requieren los siguientes pasos:

- Seleccionar una representación. En nuestro caso la representación utilizada está dada por una cadena de tamaño n (número total de tareas), la posición es el número de la tarea y su valor representa la estación a la cual la tarea ha sido asignada. Por ejemplo, la tabla 2.1 muestra un cromosoma que representa la solución de una instancia de 6 tareas repartidas en 3 estaciones. En este ejemplo las tareas 1 y 2 fueron asignadas a la estación 1, mientras que las tareas 4 y 5 se asignaron a la estación 2 y la estación 3 contiene a las tareas 3 y 6. Esta representación fue utilizada exitosamente dentro de algoritmos genéticos con anterioridad [16],[24].

1	1	3	2	2	3

Cuadro 2.3: Cromosoma utilizado para representar los individuos dentro del algoritmo evolutivo

- Decidir cómo inicializar una población. Determina el proceso de creación de los individuos para el primer ciclo del algoritmo. Normalmente, la población inicial se forma a partir de individuos creados aleatoriamente. Se puede crear también utilizando alguna técnica heurística. En cuanto al tamaño de la población hay que decir que no existen reglas fijas, normalmente una población de 25 a 100 individuos es perfectamente válida para la mayoría de los casos.

- Diseñar una forma de evaluar un individuo. La evaluación se realiza a través de una función que representa de forma adecuada el problema y tiene como objetivo suministrar una medida de aptitud de cada individuo en la población actual. Esto permite distinguir a los mejores individuos de los peores.

- Diseñar un operador de mutación adecuado. La mutación se realiza posterior al proceso de cruce, aleatoriamente se modifican uno o más genes de los individuos descendientes de la anterior generación. No es conveniente abusar del operador de mutación si no queremos que el AG se convierta en un algoritmo de búsqueda al azar. En el desarrollo de los algoritmos se realizaron diversas pruebas para ajustar el valor de la probabilidad de mutación, ubicando este valor en 0.2 en el caso de los algoritmos genéticos y 0.3 para el caso del algoritmo memético.

- Diseñar un operador de cruce adecuado. El proceso de cruce consiste en crear a los descendientes a partir del intercambio de material genético de sus padres. Existen dos técnicas principales; intercambio a partir de un solo punto de cruce y a partir de dos puntos de cruce. En ambas técnicas se obtienen dos descendientes fruto del cruce. Estas dos técnicas fueron probadas para verificar la influencia sobre los resultados del nivel de balanceo.

- Decidir cómo seleccionar los individuos para ser padres. La selección es el mecanismo por el cual soluciones más próximas al óptimo, individuos mejor adaptados, tienen mayor probabilidad de sobrevivir y ser elegidos para reproducirse. Sin embargo, también se debe dar oportunidad de reproducirse a los individuos menos buenos, ya que pueden contener material genético útil en el proceso de reproducción. Aunque existen distintos métodos para la selección de los padres, en el método desarrollado utilizamos torneo de tamaño 2, el cual consiste en hacer competir a los individuos en grupos aleatorios, normalmente parejas, el que tenga la aptitud más elevada será el ganador.

- Decidir la condición de parada. Existen varias formas de decidir en qué momento debe terminar el algoritmo; cuando se alcanza el valor óptimo, por recursos limitados de CPU, después de haber alcanzado el máximo número de evaluaciones, o bien, después de algunas iteraciones sin mejora.

2.5.3. Recocido Simulado

Recocido Simulado es un algoritmo de mejora iterativa que permite temporalmente cambios malos, que ayudan a escapar de un óptimo local. La idea es que en lugar de escoger el mejor movimiento escoja uno de manera aleatoria de la siguiente forma, si el movimiento mejora la situación, siempre se ejecuta, de otra manera el algoritmo realiza el movimiento con una probabilidad menor que 1. La probabilidad decrece exponencialmente conforme decrece la aptitud del individuo.

Un segundo parámetro T es también utilizado para determinar la probabilidad. A valores altos de T movimientos malos tienen mayor probabilidad de ser aceptados. Cuando T tiende a cero es más y más difícil que los malos movimientos sean aceptados,

hasta que el algoritmo se comporta mas o menos como el algoritmo hill climbing. El valor de T se introduce en función del número de ciclos completados. Teóricamente, si el valor de T disminuye lo suficientemente lento el algoritmo eventualmente encontrará un óptimo global. A continuación se presenta el algoritmo de Recocido Simulado tomado del AIMA.

Algoritmo 4 Algoritmo Base de Recocido Simulado

1: **función** RECOCIDO-SIMULADO ($problema, función\_T$) **regresa** una solución
entrada: *problema*, el problema a resolver
función_T, es un mapeo de la disminución de la temperatura en el tiempo.
estados: *actual*, un nodo
próximo, un nodo
T, la temperatura que controla la probabilidad de retroceder pasos
$actual \leftarrow Hacer - Nodo(Estado - Inicial[problema])$

1. **Para** $t \leftarrow 1$ hasta ∞ **hacer**

$T \leftarrow función[T]$

Si $T = 0$ **entonces regresa** *actual*

próximo $\leftarrow$ *a* sucesor de actual seleccionado aleatoriamente

$\Delta E \leftarrow Valor[próximo] - Valor[actual]$

si ΔE 0 **entonces** $actual \leftarrow próximo$

si no $actual \leftarrow$ *próimo* con probabilidad $e^{\frac{\Delta E}{T}}$

2.6. Resumen

En este capítulo se presentaron algunos trabajos de investigación desarrollados previamente para la solución del problema de balanceo de líneas de ensamble, tanto los métodos clásicos como aquellos basados en técnicas de inteligencia computacional. También se presentaron las técnicas que se proponen utilizar para el desarrollo del algoritmo como son, algoritmos de búsqueda local y algoritmos evolutivos. En los siguientes capítulos se explica el modelo de solución y el proceso por medio del cual se llegó a este modelo.

Capítulo 3

Modelo de Solución: Balanceo con Heurísticas - Algoritmo Memético - Min-Conflicts

En este capítulo se presenta el encadenamiento de heurísticas que fue desarrollado después de diversos experimentos durante la investigación. Se explica cada algoritmo utilizado dentro de este modelo de solución.

El modelo de solución para el problema de balanceo de líneas de ensamble de modelos mixtos se compone de tres etapas. Cada etapa posterior del proceso mejora el resultado obtenido de su etapa previa mediante el uso de diferentes técnicas, que fueron descritas en el capítulo anterior. En la primera etapa, se hace uso de heurísticas para crear la población inicial del algoritmo evolutivo. La siguiente etapa hace uso de un algoritmo memético. Finalmente en la última etapa del proceso se utiliza la técnica Min-Conflicts para mejorar el cumplimiento de las restricciones blandas, como se observa en la figura 3.1. Este modelo fue probado a través de diversos experimentos que serán descritos tanto en el capítulo 4 como en el capítulo 5.

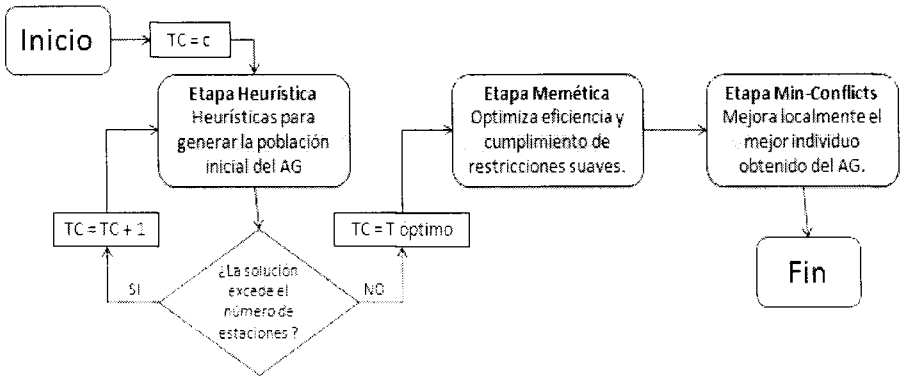

Figura 3.1: Etapas del modelo de solución

Para desarrollar el Algoritmo Memético se partió de un Algoritmo Genético debido a que numerosos estudios han demostrado la efectividad del uso de AG en la solución de problemas de optimización combinatoria, adicionalmente se comparó contra el recocido

simulado y el Algoritmo Genético. Los algoritmos evolutivos a diferencia de algoritmos como el recocido simulado, trabajan con un conjunto de soluciones a la vez. También son lo suficientemente flexibles para incluir características específicas en la codificación. El algoritmo memético tiene estas ventajas, además de otra ventaja que es el hecho de no realizar la mutación del todo a ciegas como lo hace el algoritmo genético, sino que utiliza una técnica que permite al inicio de las generaciones tener mutaciones que pudieran generar peores individuos y conforme avanza el número de generaciones decrece la posibilidad de aceptar peores individuos. A continuación se describe a detalle cada una de las etapas del modelo.

3.1. Balanceo con Heurísticas

El algoritmo 5 muestra la primera etapa del modelo propuesto. El objetivo de esta etapa es el de generar la población inicial del algoritmo memético que consiste en 50 individuos. En esta etapa se proporciona el número máximo de estaciones que pueden ser utilizadas. Este valor no se modifica en ningún momento y se considera una restricción dura del problema. Dentro de este algoritmo se incluye la ecuación 3.1 que calcula el tiempo de ciclo inicial, tomará el máximo valor entre el tiempo de ciclo deseado y la sumatoria de los tiempos de cada tarea por cada modelo dividido entre el número de estaciones disponibles, tal como si calculáramos el tiempo de ciclo óptimo. A continuación se lista la información que debe ser proporcionada para comenzar esta etapa.

Información proporcionada al inicio del algoritmo:

- Número de productos (M) distintos que se ensamblarán en la línea.
- Demanda esperada para cada producto (q_m); la sumatoria de los valores q para cada producto deberá ser igual a uno.
- Número de operaciones (n) a realizar sobre los productos, deberá ser la misma cantidad de operaciones para todos ellos. Para el caso en el cual algún producto no contenga una operación que si es necesaria para el resto de los productos, el tiempo de operación se colocará en cero.
- Tiempo de ciclo (Tc) de las estaciones, es el tiempo transcurrido entre dos artículos. Al inicio se proporciona un tiempo de ciclo deseado c, aunque usualmente el tiempo de ciclo está relacionado con la cantidad de productos a fabricar y el tiempo disponible para fabricarlos. Sin embargo, no lo podemos calcular únicamente dividiendo estos valores ya que pudieran existir retrasos debidos a fallas de equipo o errores humanos, por lo cual es importante definir este parámetro.

- Tiempo de operación de cada tarea para cada modelo m (t_{im}). Este tiempo puede variar entre los diferentes modelos, es por ello que es necesario especificarlo para cada uno.
- Número de estaciones (k) en la línea de ensamble.
- Grafo de precedencia de tareas para los modelos. Usualmente se concentran en un solo grafo las relaciones de precedencia de todos los modelos.

Algoritmo 5 Modelo de Solución: Balanceo con Heurísticas

1: **Balanceo con Heurísticas**
Procedimiento:

1. Proporcionar los valores para los parámetros c, qm , número de operadores o estaciones k, así como los datos de las tareas que serán asignadas.
2. Obtener el tiempo de ciclo que se utilizará como punto de partida para las asignaciones.

$$LB = max_{m=1,...M} \left\{ \left[\sum_{i=1}^{N} t_{im}/k \right], c \right\} \tag{3.1}$$

Donde:

t_{im}: Tiempo de procesamiento de la tarea i en el producto m.
k: Número de estaciones.
c: Tiempo de ciclo inicial o deseado.

3. Incrementar el Tc, como en un principio, el número de operadores resultantes con el tiempo de ciclo dado puede ser mayor al permitido, se incrementa el tiempo de ciclo en una unidad de tiempo cada vez que se termina el proceso de asignación de tareas hasta encontrar un Tc que cumpla esta restricción de operadores.
4. Fijar el Tc, una vez que se haya obtenido una solución válida con el menor tiempo de ciclo posible, se fijan los parámetros tanto del tiempo de ciclo como del número de operadores hasta completar 50 resultados de asignación válidos. Estos 50 elementos son los que formarán la población inicial del algoritmo evolutivo en la siguiente etapa. En el proceso de asignación de tareas se tienen los pasos descritos en el algoritmo 6.

Dentro del algoritmo 6 se mencionaron tres heurísticas que se utilizan para hacer la selección de la tarea que sería asignada a la estación actual. Como se indica un número aleatorio entre 1 y 3, establece la heurística a utilizar:

Algoritmo 6 Asignación de Tareas

1. Crear tantas estaciones, como operadores permitidos se tengan.
2. Generar un número al azar entre 1 y 3, para seleccionar la tarea que será asignada a la estación actual mediante alguna de las tres heurísticas disponibles, comenzando con la tarea 1.
3. Verificar que se cumplan las restricciones de precedencia antes de asignar la tarea, si no se cumplen esta tarea no es asignada, y se elige una nueva.
4. Verificar si la estación actual aún tiene tiempo disponible para realizar esta tarea.

   Si es posible, verificar si existe incompatibilidad con tareas ya asignadas a la estación actual, si no existe, asignar la tarea y si existe asignar la tarea a la siguiente estación.

   Si la estación no tuvo tiempo disponible, crear una nueva estación y simplemente asignar la tarea a la nueva estación.

- Si el resultado es 1: selecciona la tarea en función del máximo número de sucesores.
- Si el resultado es 2: selecciona la tarea tomando en cuenta el máximo tiempo promedio de procesamiento. Este procedimiento es semejante al algoritmo de Moodie y Young [20], mencionado en el capítulo anterior.
- Si el resultado es 3: la tarea se selecciona tomando en cuenta el máximo peso posicional. Para obtener el peso posicional se calcula la suma de su tiempo más los tiempos de todas las tareas que le siguen. En este caso el procedimiento es similar al algoritmo de Hegelson y Birnie 1, también descrito en el capítulo anterior.

3.2. Algoritmo Memético

El objetivo en esta etapa es mejorar el nivel de balanceo de las estaciones, y también mejorar el cumplimiento de las restricciones suaves o preferencias del usuario. En esta etapa, el tiempo de ciclo de cada individuo puede variar. El algoritmo 7 describe los pasos principales de esta etapa, se observa que la función principal que se desea maximizar es la ecuación 3.2 que tiene dos partes, la eficiencia y la violación de restricciones suaves. En cuanto a la eficiencia, se toma la sumatoria de las diferencias que

existen entre el tiempo de ciclo óptimo menos el tiempo actual de cada estación que al dividir entre el total de tiempo disponible se normaliza el valor. A continuación se describen los operadores de selección, cruce y mutación utilizados.

Selección

La selección por torneo de tamaño 2, consiste en comparar uno de los individuos en la población contra el individuo siguiente, aquél que tenga la mejor evaluación de la función objetivo gana el torneo y será seleccionado. Los individuos deben estar ordenados aleatoriamente, esta comparación se realiza con cada uno de los individuos de la población.

Cruce de un punto

El cruce es el principal operador del algoritmo genético. Esta operación recombina pedazos de información de diferentes individuos, se eligen dos padres para hacer el cruce a partir de ellos se obtienen dos nuevos individuos. Para realizar el cruce se siguen los pasos establecidos en el algoritmo 8 hasta terminar de hacer cruce con toda la población hasta obtener 50 individuos.

Mutación

Este operador se lleva a cabo después del cruce y altera aleatoriamente la información genética de los cromosomas. Para realizar la mutación se siguen los pasos descritos en el algoritmo 9 según la probabilidad de mutación hasta terminar con toda la población.

3.3. Min-Conflicts

De la etapa anterior obtuvimos al mejor individuo, que nos servirá en esta última etapa para tratar de mejorarlo localmente. El algoritmo 10 muestra los pasos para llevar a cabo la etapa utilizando la técnica Min-Conflicts.

Para calcular la aptitud en este algoritmo también se utiliza la función objetivo definida en la etapa del algoritmo memético, en la ecuación 3.2. En lo referente a las restricciones suaves, está pensado para mejorar la situación con respecto a las violaciones de las restricciones ergonómicas que pudieran surgir en la distribución de tareas en las estaciones.

Cuando se trabaja en una línea de ensamble los principales problemas ergonómicos que pueden surgir son aquellos que se derivan de la postura en general del cuerpo, la rotación de la muñeca y el antebrazo, la flexión o desviación de la muñeca, la presión de los dedos o de la mano, o bien las debidas al levantamiento de cargas. Aunque hasta

Algoritmo 7 Modelo de Solución: Etapa Algoritmo Memético

1: **Etapa Algoritmo Memético**
Procedimiento:

1. Proporcionar valores de probabilidad de cruce *(Pc)* y probabilidad de mutación *(Pm)*.

2. Calcular el tiempo de ciclo óptimo *(to)* utilizando la ecuación 3.4.

3. Inicializar la población del algoritmo memético utilizando las soluciones obtenidos de la etapa previa de balanceo con heurísticas.

4. Definir la temperatura inicial *(Ti)* para el proceso de mutación mediante recocido simulado, el número de ciclos será igual al número de generaciones y el decremento se hará proporcionalmente hasta llegar a una temperatura de 0.

5. Durante x generaciones, realizar los siguientes pasos.

   a) Realizar selección de padres mediante torneo con tamaño de torneo 2.

   b) Realizar cruce, recombinación de padres mediante cruce de 1 punto.

   c) Realizar mutación utilizando Recocido Simulado de acuerdo a la *Pm*.

   d) Evaluar la siguiente función objetivo:

$$B = (eficiencia * 0.8) + (vrs * 0.2) \tag{3.2}$$

   donde:

$$eficiencia = 1 - \frac{\sum_{i=1}^{k} abs\langle to - \sum_{m=1}^{M} q_m S_{km}\rangle}{to * k} \tag{3.3}$$

$$vrs = 1 - \frac{estaciones\_con\_violación\_en\_restricciones\_suaves}{número\_total\_de\_estaciones} \tag{3.4}$$

$$to = max_{m=1,...M} \left\{ \sum_{i=1}^{n} t_{im}/k \right\} \tag{3.5}$$

6. Disminuir la temperatura en cada generación.

7. Devolver el mejor individuo encontrado y su tiempo de ciclo.

Algoritmo 8 Operador de cruce de un punto

1: **Cruce de un punto**
Procedimiento:

1. Seleccionar dos padres dentro del *pool* de apareamiento y generar un punto de cruce al azar.

2. Tomar las tareas de la estación 1 hasta la estación definida por el punto de cruce del padre 1 para construir el nuevo cromosoma, y tomar las tareas restantes de la siguiente estación hasta el final del padre 2.

3. Si quedan tareas sin asignar:

   a) Sacar tareas precedentes y consecuentes de cada tarea no asignada.

   b) Determinar entre que estaciones es posible asignar esta tarea de acuerdo a la asignación de sus tareas precedentes y consecuentes, así como tomando en cuenta las restricciones de incompatibilidad de tareas.

   c) Seleccionar en que estación ubicar la tarea mediante la evaluación de las estaciones donde es posible asignar esta tarea en función del tiempo disponible de cada estación.

4. Para construir el segundo descendiente, tomar las tareas de la estación 1 hasta la estación definida por el punto de cruce del padre 2 para construir el nuevo cromosoma, y tomar las tareas restantes de la siguiente estación hasta el final del padre 1.

5. Si quedan tareas sin asignar, éstas se reubican como en el paso 3.

6. Verificar nuevamente que no se excedan los tiempos de ciclo en cada una de las estaciones y reasignar tareas que se puedan reasignar para que las estaciones queden dentro de tiempo permitido.

7. Evaluar la aptitud a los nuevos individuos con la función.

8. Si la aptitud de los nuevos individuos es mejor que el peor de la población anterior reemplazar a los individuos padres por los nuevos individuos de lo contrario los padres permanecerán en la población.

Algoritmo 9 Mutación con Recocido Simulado

1: **Mutación con Recocido Simulado**

Procedimiento:

Para cada individuo de la población, realizar los siguientes pasos:

1. Mientras la temperatura no sea cero o el valor óptimo de balanceo no se haya encontrado y la probabilidad de mutación lo permita realizar los siguientes pasos:

   - Durante 5 veces realizar:

     a) Elegir aleatoriamente una tarea.

     b) Verificar todos los movimientos posibles hacia nuevas estaciones para la tarea elegida.

     c) Elegir aleatoriamente uno de esos movimientos para obtener el nuevo individuo.

     d) Para llevar a cabo la nueva asignación verificar si $T = 0$, devolver el individuo actual

     Si no, calcular Δ.
     $\Delta = B$ del nuevo individuo - B del individuo actual.
     Donde B es calculado con la ecuación 3.2

     Si $\Delta > 0$ entonces el individuo actual = nuevo individuo.

     Si no, con una probabilidad $p = exp(\frac{\Delta}{T})$ aceptar o no el nuevo individuo.

2. Regresar el último individuo para colocarlo dentro de la próxima generación.

Algoritmo 10 Modelo de Solución - Etapa Min-Conflicts

1: **Etapa Min-Conflicts Procedimiento:**
Durante 200 iteraciones realizar los siguientes pasos:

1. Sacar estaciones con violación en restricciones suaves.

   a) Determinar cuales estaciones tienen conflicto dentro de ellas.

   b) Determinar cuales tareas son las conflictivas.

2. Elegir aleatoriamente en cuál estación se tratará de resolver el conflicto.

3. Elegir aleatoriamente la tarea que se intentará reubicar dentro de esa estación.

4. Definir el rango de estaciones en las que es posible reubicar la tarea, tomando en cuenta restricciones de precedencia e incompatibilidad de tareas.

5. Para cada estación donde es posible reubicar dicha tarea:

   a) Calcular la aptitud del individuo.

   b) Verificar nuevamente las restricciones suaves violadas.

6. Si existen asignaciones que mejoraron el número de restricciones suaves violadas:

   a) Elegir de todas las asignaciones que fueron factibles, aquellas que mejoraron el número de restricciones suaves violadas.

   b) Calcular la nueva aptitud y comparar el mayor valor de aptitud con el valor anterior, se espera que como máximo se empeore 10 % y entonces esa asignación es el nuevo individuo de lo contrario permanece el individuo anterior.

7. Si no existen asignaciones que mejoren el número de restricciones suaves violadas pero hay algunas que lo dejaron igual:

   Verificar si mejora la aptitud en alguna de ellas y este será el nuevo individuo, de lo contrario, también permanece el anterior.

8. Finalmente en caso de que todas las asignaciones hayan empeorado las restricciones suaves violadas, no se admite la asignación y permanece la solución que ya se tenía.

el momento sólo existen recomendaciones en cuanto al límite de tiempo o repeticiones máximas permitidas durante la jornada de trabajo, en nuestro problema de estudio únicamente calculamos las estaciones que contienen restricciones suaves violadas tomando en cuenta criterios de repetitividad.

Cada tarea contiene una bandera que nos dice, que si se acumulan varias tareas de ese tipo en una misma estación nos podría causar un problema ergonómico. ¿Cuál es el límite de tareas permitidas por estación?, ese es un dato que lo debe decidir el usuario. En el presente trabajo no se exponen las recomendaciones ergonómicas ya que existen numerosos estudios dedicados exclusivamente a este tema, entre ellos se encuentran [23], [8], [29], [31].

Para realizar el conteo de las estaciones con violaciones en restricciones suaves se realiza lo siguiente, para cada estación:

1. Contar el número de tareas que de acumularse pudieran ocasionar un problema en la estación.

2. Si el número de tareas supera el valor definido como máximo entonces esa estación tiene un conflicto, de lo contrario no existe ningún conflicto en esa estación.

Al terminar con todas las estaciones, se cuenta el número de estaciones que tienen conflictos en ellas.

3.4. Resumen

En este capítulo se describió el modelo de solución desarrollado para el problema de balanceo de líneas de ensamble de modelo mixto. Después de haber realizado diversos experimentos, que se describen a detalle en los siguientes capítulos, se obtuvo el mejor encadenamiento de heurísticas que como se describió consiste en tres etapas. La primera de ellas incluye un balanceo con heurísticas que utiliza técnicas desarrolladas en los primeros años de investigación sobre balanceo de líneas de producción. La segunda etapa contempla el uso de un algoritmo memético cuya mutación se realiza mediante un algoritmo de recocido simulado y finalmente la última etapa utiliza un algoritmo basado en la técnica de búsqueda local min-conflicts.

Capítulo 4

Optimización del Balanceo de Líneas de Ensamble de Modelo Mixto Mediante Algoritmos Genéticos

En este capítulo se describen experimentos preliminares utilizando tres versiones de un algoritmo genético que fue implementado para resolver el problema de balanceo. Los experimentos están enfocados en verificar el desempeño de dicho algoritmo. También se describe un generador de problemas utilizado para generar instancias del problema de balanceo de líneas. Estas instancias fueron utilizadas para realizar los experimentos que se tomaron como base para la determinación de la técnica principal y su afinación, para garantizar los mejores resultados de dichos algoritmos en la solución al problema de balanceo de líneas de ensamble. Se realizaron pruebas enfocadas a la afinación del algoritmo genético, estos algoritmos fueron codificados en Matlab.

En el modelo de solución descrito en el capítulo anterior se detallan dos procesos que se desarrollan como etapa final, esta parte será considerada durante el siguiente capítulo, de manera que durante esta primera parte nos centraremos únicamente en las primeras dos etapas del algoritmo desarrollado.

Para llegar al algoritmo descrito en la sección anterior se realizaron varias modificaciones, incluyendo utilizar el cruce de dos puntos que consiste en lo siguiente:

1. Seleccionar dos individuos y dos puntos de cruce al azar.

2. Tomar las tareas de la estación 1 hasta la estación definida por el punto de cruce más cercano, y del segundo punto de cruce al final de las estaciones, del padre 1 para construir el nuevo cromosoma, después tomar las tareas restantes del punto de cruce 1 al punto de cruce 2 del padre 2.

3. Si quedan tareas sin asignar:

    a) Sacar tareas precedentes y consecuentes de cada tarea no asignada.

    b) Verificar entre que estaciones es posible asignar esta tarea de acuerdo a la asignación de sus tareas precedentes y consecuentes, así como tomando en cuenta las restricciones de incompatibilidad de tareas.

c) Seleccionar en que estación ubicar la tarea mediante la evaluación de las estaciones donde es posible asignar esta tarea en función del tiempo disponible de cada estación.

4. Para construir el segundo descendiente tomar las tareas de la estación 1 hasta la estación definida por el punto de cruce mas cercano, y del segundo punto de cruce al final de las estaciones, del padre 2 para construir el nuevo cromosoma, después tomar las tareas restantes del punto de cruce 1 al punto de cruce 2 del padre 1.

5. Si quedan tareas sin asignar realizar paso 3.

6. Verificar nuevamente que no se excedan los tiempos de ciclo en cada una de las estaciones y reasignar tareas que se puedan reasignar para que las estaciones queden dentro de tiempo permitido.

7. Evaluar a los nuevos individuos con la función.

8. Si la aptitud de los nuevos individuos es mejor que el peor de la población anterior reemplazar a los individuos padres por los nuevos individuos de lo contrario los padres permanecerán en la población.

4.1. Algoritmo Genético Implementado

Para llevar a cabo esta primera etapa de experimentación se implementaron tres versiones del algoritmo genético y un generador de problemas. Al final de los experimentos tomamos uno de estos algoritmos como base para continuar desarrollándolo en la siguiente etapa. A continuación se describen cada uno de estos algoritmos implementados y posteriormente el generador de problemas.

4.1.1. Versión 1: Algoritmo Genético con Disminución de Tiempo de Ciclo por Generación

Este algoritmo está compuesto por dos etapas: la primera etapa permanece igual a la primera etapa del modelo de solución descrito en el capítulo anterior, incluso las otras dos soluciones implementadas para los experimentos preliminares utilizan la misma etapa heurística, la cual esta descrita en el algoritmo 5. La etapa del Algoritmo Genético se describe en el algoritmo 11. En esta etapa existen algunas diferencias entre las principales se encuentra el uso de una función objetivo diferente, un tanto mas elaborada ya que teóricamente nos proporciona una combinación del nivel de balanceo entre cada estación y también el nivel de balanceo entre cada modelo. Esta función

objetivo fue tomada del trabajo de Vilarinho y Simaria [16].

Como se había mencionado anteriormente la función objetivo utilizada en el algoritmo genético, toma en cuenta dos aspectos, la primera parte con la fórmula 4.2 toma en cuenta el nivel de balanceo entre las estaciones y el valor obtenido de la fórmula 4.5 toma en cuenta el nivel de balanceo entre los modelos, éstas funciones objetivo fueron una combinación de funciones objetivo utilizadas en otro trabajo de investigación, el de Simaria y Vilarinho en [16]. Otra de las diferencias importantes existentes en este algoritmo con respecto al modelo es que el tiempo de ciclo va disminuyendo, comenzando por una unidad abajo del obtenido de la etapa heurística, sin importar que en un momento dado al final de la etapa, no exista ningún individuo con este tiempo de ciclo. En cuanto al cruce y la selección se realiza de la misma manera que se describió en el modelo de solución. El operador de mutación por su parte, como este es un algoritmo genético se realiza de manera diferente, como se describe a continuación.

Mutación

La mutación altera aleatoriamente la información genética de los cromosomas. Para realizar la mutación se siguen los siguientes pasos hasta terminar de hacer mutación con toda la población:

1. Cada individuo se muta de acuerdo a la probabilidad de mutación proporcionada.
2. Si resultó en que el individuo se muta, seleccionar aleatoriamente un pequeño conjunto de tareas, como valor máximo será el 10 % del total de tareas.
3. Reasignar estas tareas de la misma forma que lo hace el algoritmo de cruce en el paso 3.

4.1.2. Versión 2: Algoritmo Genético con Balanceo por Modelos

Este algoritmo está compuesto también por dos etapas, la etapa heurística que genera la población inicial permanece igual a la etapa heurística del modelo de solución descrito en el capítulo anterior. La etapa del Algoritmo Genético se describe en el algoritmo 12, en esta etapa también existen algunas diferencias con respecto al modelo de solución final y con respecto al algoritmo anterior. Entre las principales diferencias se encuentra el uso de una sola función objetivo, teóricamente esperábamos que si existía un buen balanceo entre los modelos como resultado existiría un buen balanceo entre las estaciones, es por ello que se utilizó únicamente la parte de la función objetivo que hace referencia a este aspecto del balanceo. En este caso tampoco trata de reducir el

Algoritmo 11 Versión 1 - Algoritmo Genético con Disminución de Tiempo de Ciclo por Generación

1. Proporcionar valores para las variables Pc, Pm e inicializar población del AG.
2. Verificar si en la población aún existen elementos factibles con el Tc actual, mientras exista al menos un cromosoma válido realizar los siguientes pasos:

   a) Disminuir en una unidad de tiempo el tiempo de ciclo.

   b) Realizar selección mediante torneo con tamaño de torneo 2.

   c) Realizar cruce.

   d) Realizar mutación.

   e) Evaluar la siguiente función objetivo:

$$UF = 0.2 * (U) + 0.8 * (F) \tag{4.1}$$

   donde:

$$\text{LL} = \text{Longitud de línea.}$$

$$U = \frac{LL}{LL-1} \sum_{k=1}^{LL} \left[\frac{\sum_{m=1}^{M} q_m S_{km}}{IT} - \frac{1}{LL} \right]^2 \tag{4.2}$$

$$S_{km} = \sum_{k=1}^{LL} abs(Tc - \sum_{m=1}^{M} (q_m * t_{im})) \tag{4.3}$$

$$IT = \sum_{k=1}^{LL} \sum_{m=1}^{M} q_m S_{km} \tag{4.4}$$

$$F = 1 - B \tag{4.5}$$

$$B = \frac{M}{LL(M-1)} \sum_{K=1}^{LL} \sum_{m=1}^{M} \left(S_{km} - \frac{1}{M} \right)^2 \tag{4.6}$$

   f) Contar operadores utilizados en cada cromosoma y si no existe algúno que cumpla la restricción del número de operadores, realizar del paso 2 al 6 durante 20 veces o hasta encontrar al menos un individuo válido.

3. Devolver el mejor individuo encontrado y su tiempo de ciclo actual.

tiempo de ciclo, el cual permanece igual hasta el final de esta segunda etapa. Esto es debido a que la función objetivo es independiente del tiempo de ciclo.

4.1.3. Versión 3: Algoritmo Genético con Tiempo de Ciclo Variable por Individuo

Este algoritmo también está compuesto por dos etapas, la etapa heurística de igual forma a la etapa heurística del modelo de solución. Esta segunda etapa también es un Algoritmo Genético, en el se ha cambiado la función objetivo para medir la eficiencia en general de la línea. Sin embargo el tiempo de ciclo utilizado para realizar los cálculos es variable, cada individuo revisa los tiempos que tiene en cada estación y el tiempo de ciclo será el máximo tiempo registrado en las estaciones. De esta manera el valor de la eficiencia es mas real a las condiciones de cada individuo, al contrario de que fuera tomado un mismo tiempo de ciclo para toda la población durante todas las generaciones. El algoritmmo 13 describe el procedimiento para la etapa del algoritmo genético.

4.2. Generador de Problemas

A continuación se describe un generador de problemas que se implementó para obtener instancias del problema de balanceo de líneas, con el objetivo de llevar a cabo experimentos utilizando estas instancias. Se asegura que los problemas generados tienen solución y que además pueden tener una solución óptima, como el hecho de no tener tiempo muerto en las estaciones para un tiempo de ciclo dado. Como parámetros de entrada se delimitaron los siguientes:

- Número de tareas (n) por estación, esto es aproximado, ya que al tener los tiempos de las operaciones diferentes, controlar el número de tareas en cada estación se va complicando.
- Número de modelos (M) que se ensamblarán en la línea.
- Tiempo mínimo que puede tomar cada tarea ($Tmin$), este tiempo es de referencia, ya que al hacer los ajustes en las tareas para obtener un balanceo perfecto, estos tiempos pueden variar.
- Tiempo máximo que puede tomar cada tarea ($Tmax$): El tiempo máximo y el tiempo mínimo se refiere a los rangos entre los cuales pueden variar los tiempos de procesamiento de cada tarea.

Algoritmo 12 Versión 2 - Algoritmo Genético con Balanceo por Modelos

1: **ETAPA Algoritmo Genético**
Procedimiento:

1. Proporcionar valores de las variables Pc y Pm.

2. Utilizando los individuos obtenidos de la etapa anterior o etapa heurística, inicializar la población del algoritmo genético.

3. Mientras la aptitud actual sea mejor que la mejor aptitud durante el experimento, realizar los siguientes pasos, en caso contrario terminar y devolver resultados.

   a) Actualizar el mejor valor de aptitud encontrado.

   b) Realizar selección mediante torneo con tamaño de torneo 2.

   c) Realizar cruce.

   d) Realizar mutación.

   e) Evaluar la siguiente función objetivo:

$$F = 1 - B \tag{4.7}$$

$$B = \frac{M}{LL(M-1)} \sum_{K=1}^{LL} \sum_{m=1}^{M} \left(S_{km} - \frac{1}{M} \right)^2 \tag{4.8}$$

   f) En caso de que la aptitud de los individuos no esté mejorando, realizar los pasos 2 al 5 durante 10 veces o bien hasta que mejore la aptitud actual.

4. Devolver la última población que se tiene y el tiempo de ciclo actual.

Algoritmo 13 Versión 3: Algoritmo Genético con Tiempo de Ciclo Variable por Individuo

1: **ETAPA Algoritmo Genético**
Procedimiento:

1. Proporcionar valores de las variables Pc y Pm.
2. Utilizando los individuos obtenidos de la etapa anterior, inicializar la población del algoritmo genético.
3. Durante 300 generaciones realizar los siguientes pasos.
   a) Realizar selección mediante torneo con tamaño de torneo 2.
   b) Realizar cruce.
   c) Realizar mutación.
   d) Evaluar la siguiente función objetivo:

$$TB = 1 - TBv \tag{4.9}$$

$$TBv = \frac{\sum_{k=1}^{LL}(Tc - \sum_{m=1}^{M} q_m S_{km})}{Tc * LL} \tag{4.10}$$

4. Devolver el mejor individuo encontrado y su tiempo de ciclo actual.

- Número de estaciones (k): Dado que en los problemas reales tenemos un número limitado de estaciones, este parámetro durante la solución del problema se considerará una restricción dura.

Estos parámetros son indispensables para obtener un problema propuesto, sin embargo tenemos también otros parámetros que son útiles para obtener el problema.

- Máximo número de antecedentes para las tareas (PR).
- Probabilidad de tener tareas antecedentes (Pa). Entre mayor sea la probabilidad más serán las restricciones de precedencia para esa tarea en particular. Sin embargo el que estén más restringidas no necesariamente significa que el problema sea mas difícil ya que como se había explicado en el capítulo 2 cuando un nodo está conectado por un solo arco el problema es muy sencillo ya que no hay más opciones de solución para escoger.
- Probabilidad de tener tareas incompatibles (Pi). En este caso también entre mayor sea la probabilidad, la instancia estará más restringida.
- Probabilidad de tener diferentes tiempos para los modelos (pdt), esto es para evitar que el tiempo de una operación sea el mismo para todos los modelos, ya que esto resultaría en tener un problema de balanceo de un solo modelo.

Los datos que devuelve el generador son los siguientes:

1. Una matriz con los datos de las operaciones, como ejemplo se muestra la tabla 4.1. En esta tabla se observan siete columnas en total. La primera columna corresponde al número de la tarea, las siguientes dos columnas corresponden a los tiempos de ejecución de la tarea para cada modelo, M1 es el tiempo que toma realizar cada tarea al modelo 1 y M2 se refiere al tiempo de la misma tarea pero en el modelo 2. Las columnas 4, 5 y 6 representan las restricciones de precedencia que tiene cada tarea. En total, cada tarea puede tener hasta tres relaciones de precedencia, esto dependerá de la probabilidad que hayamos escrito como entrada. La última columna se refiere a la incompatibilidad de tareas, en este ejemplo no aparecen tareas incompatibles, también depende de la probabilidad que hayamos indicado al generador de problemas.
2. Distribución óptima del cuadro 4.2. Se refiere a cómo debería quedar la distribución de las tareas en las estaciones para obtener un balanceo óptimo; con esto nos aseguramos que realmente existe un balanceo perfecto al que deberíamos llegar cuando probamos cada algoritmo.

Cuadro 4.1: Ejemplo de instancia proporcionada por el generador de problemas

Tareas	Tiempo		PR			I.
	M1	M2				de tareas
1	37	48	5	0	0	0
2	35	45	3	7	8	0
3	38	27	0	0	0	0
4	33	23	7	5	0	0
5	35	35	0	0	0	0
6	32	32	1	3	0	0
7	26	26	5	0	0	0
8	39	39	0	0	0	0
9	25	25	7	8	0	0

Cuadro 4.2: Distribución óptima de la instancia proporcionada por el generador de problemas

2	3	2	3	1	3	1	1	2

La siguiente figura 4.1 muestra como queda la carga de trabajo en cada estación utilizando la distribución óptima. En este caso existen tres tareas en cada estación de la solución óptima. Las tareas 5, 7 y 8 deben estar en la estación 1, las tareas 1, 3 y 9 deben estar en la estación 2 y las tareas restantes pertenecen entonces a la estación 3. Con esta distribución la suma de los tiempos nos da como resultado 100 en cada una de las estaciones.

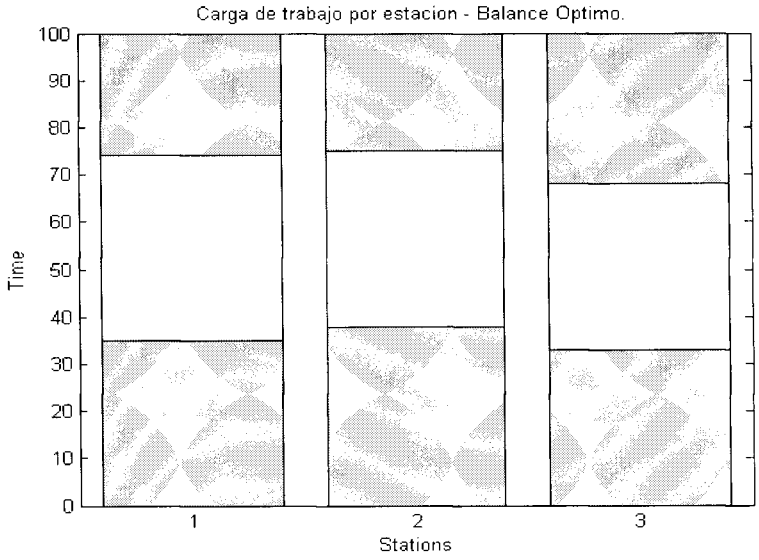

Figura 4.1: Ejemplo de gráfica de la carga de trabajo por estación

4.3. Experimentos con Algoritmos Implementados.

Se realizaron experimentos utilizando dos problemas que fueron obtenidos con ayuda del generador de problemas, en ambos casos el balanceo, debería llegar al valor óptimo 1. Las características de los problemas son los siguientes:

Problema 1:

Número de estaciones $(k) = 8$

$q_m = [0.50.5]$

$c = 90$

$M = 2$

Número de tareas $(n) = 35$

Tiempo de ciclo óptimo = 100

West ratio = 4.375

Radio de variabilidad de tiempo = 10

La diferencia entre el problema 1 y el problema 2 recae principalmente en el número de tareas utilizadas en cada problema, en el segundo problema se observa también una mayor variabilidad de los tiempos de cada tarea, por ejemplo los valores pueden ser tan pequeños como 5 o tan grandes como 74, mientras que en el problema 1 se concentran los valores entre 20 y 30 unidades de tiempo, esto depende del promedio de tareas que caben por estación, *west ratio*, y el tiempo de ciclo óptimo.

Problema 2:

Número máximo de operadores permitidos (número de estaciones): 10

q_m = [0.50.5]

MRT = 140

M = 2

Número de tareas = 100

Tiempo de ciclo óptimo = 150

West ratio = 10

Radio de variabilidad de tiempo = 74

4.3.1. Combinación de Funciones Objetivo para Algoritmo Genético Versión 1

Este experimento previo se realizó debido a que este algoritmo mezcla dos funciones objetivo para saber en que porcentaje era conveniente mezclar cada función objetivo, se verificó con cada valor comenzando con 0.1 para el valor U de la ecuación 4.2 y 0.9 para el valor F de la ecuación 4.5, en este caso tomamos como referencia el nivel de balanceo entre las estaciones delimitado por la ecuación 4.5. Los resultados se observan en el cuadro 4.3, el problema utilizado en este experimento fue el problema 1.

Se realizaron 50 repeticiones con cada combinación de funciones objetivo, en la tabla 4.3 se muestra resultados promedios del balanceo obtenido con cada mezcla, así como también la desviación estándar en cada caso.

Cuadro 4.3: Balanceo obtenido con diferente combinación de funciones objetivo en la versión 1 del algoritmo genético

Porcentajes F	0.9 F	0.8 F	0.7 F	0.6 F	0.5 F	0.4 F	0.3 F	0.2 F	0.1 F
Porcentajes U	0.1 U	0.2 U	0.3 U	0.4 U	0.5 U	0.6 U	0.7 U	0.8 U	0.9 U
Nivel de Balanceo	84.56	90.79	82.78	70.84	90.07	73.97	73.02	71.19	72.74
Desviación estándar	0.049	0.078	0.021	0.042	0.09	0.028	0.043	0.046	0.015

Obtenemos un mejor nivel de balanceo cuando la función objetivo, la calculamos con la mezcla de porcentajes de 0.8 para el valor de F y 0.2 en el valor de U, es decir cuando le damos mayor peso al balanceo entre estaciones, más allá del balanceo entre los modelos generados por el valor de U.

4.3.2. Experimento para Determinar el Procedimiento de Cruce a Utilizar

En la etapa del Algoritmo Genético de las versiones implementadas se llevó a cabo una modificación durante el proceso de cruce de los individuos: se añadió una restricción que consiste en lo siguiente, el procedimiento permite reemplazar a los padres por los nuevos individuos solamente si la aptitud de éstos es mejor que el peor observado en la población anterior. De otra manera, si no colocamos esta restricción los nuevos individuos se aceptan siempre y los padres son reemplazados en cualquier caso. En ambos casos tenemos inconvenientes, en el primer caso el algoritmo podría converger tan rápido que se atore en un óptimo local, y en el segundo caso podríamos perder información de algún buen individuo. Se llevó a cabo 100 corridas utilizando los tres algoritmos implementados, los resultados de balanceo promedio y desviación estándar se muestran en el cuadro 4.4.

Se observa que en general se comporta ligeramente mejor al colocar esta restricción a los algoritmos, debido seguramente a que cuando acepta los individuos con menor aptitud explora un mayor espacio de búsqueda que influye en que no mejore tanto como en el caso contrario, ya que lo va delimitando a mejorar cada vez más los individuos de la población. Es importante recordar que el máximo valor que se puede obtener es 1, en este caso los tres algoritmos están obteniendo muy buenos resultados.

Cuadro 4.4: Balance del mejor encontrado del algoritmo genético con diferente procedimiento de cruce

Nivel de balanceo	Versión 1	Versión 2	Versión 3
Cruce con restricción	0.9414	0.9744	0.9718
Desviación estándar	0.038	0.139	0.021
Cruce sin restricción	0.9464	0.9451	0.9413
Desviación estándar	0.042	0.048	0.014

4.3.3. Experimento Comparación Cruce de 1 y 2 Puntos

Para realizar este experimento se tomó en cuenta lo obtenido del anterior, los algoritmos conservaron la restricción de aceptar los nuevos individuos solo en caso de que la aptitud sea mejor que el peor observado en la población anterior. Se continuó con el mismo problema 1, pero este experimento ahora se incluyó el método de cruce de dos puntos descrito al inicio de éste capítulo. De igual forma se realizaron 100 pruebas para verificar el comportamiento del balanceo de los individuos. En la tabla 4.5 se muestran los resultados obtenidos de este experimento, el nivel de balanceo promedio obtenido y la desviación estándar.

Cuadro 4.5: Resultados de la comparación entre cruce de 1 y 2 puntos

Nivel de balanceo	Versión 1	Versión 2	Versión 3
Cruce 1 punto	0.9414	0.9744	0.9718
Desviación estándar	0.042	0.048	0.01
Cruce 2 puntos	0.9402	0.9744	0.966
Desviación estándar	0.058	0.069	0.046

No se observa una fuerte diferencia entre realizar el cruce de un punto y de dos puntos, sin embargo el procedimiento de realizar cruce de dos puntos es más largo, por lo cual toma más tiempo llevarlo a cabo. Al final de este experimento se decide continuar con cruces de un solo punto ya que el de dos puntos no nos ofreció ninguna ventaja.

4.3.4. Comparativa General de los Algoritmos

Después de llevar a cabo los experimentos anteriores y haber decidido entre que tipo de cruce, como debía mezclarse la función objetivo y las restricciones del algoritmo genético en todos los algoritmos implementados, finalmente se probaron en ambos problemas, para verificar el balanceo obtenido en cada caso, así como los tiempos de ejecución de cada uno. El cuadro 4.6 concentra el nivel de balanceo obtenido de cada algoritmo en cada problema. Y el cuadro 4.7 muestra los tiempos de ejecución.

Cuadro 4.6: Comparativa general de las versiones del algoritmo genético implementado

Nivel de balanceo	Versión 1	Versión 2	Versión 3
Problema 1	0.9414	0.9744	0.9718
Problema 2	0.7758	0.8297	0.9721

Cuadro 4.7: Tiempos de ejecución

	Problema 1 (s)	Problema 2 (s)
Versión 1	64.31	128.43
Versión 2	41.96	86.72
Versión 3	64.44	63.99

En ambos casos la versión 3 se comporta mejor, obtiene buenos resultados tanto para el problema largo como para el problema corto, y en cuanto al tiempo de ejecución también obtiene buenos resultados, sobre los otros dos.

El algoritmo genético versión 1, con disminución de tiempo de ciclo por generación, al utilizar el mismo tiempo de ciclo para todos los individuos perdía buenos individuos que tenían un tiempo de ciclo menor. Por otra parte el algoritmo genético versión 2, con balanceo por modelos, era independiente del tiempo de ciclo, pero no balanceaba correctamente entre las estaciones ya que no tomaba en cuenta este aspecto y como resultado no lograba un buen balanceo. Sin embargo la versión 3 del algoritmo genético, en su función de optimización contempla la eficiencia que toma en cuenta las estaciones y el tiempo de ciclo del individuo, y al ser variable para cada uno identifica mejor a los buenos individuos dentro de la población.

Cabe mencionar que fue mas utilizado el problema 1 durante la experimentación ya que al ser un problema pequeño nos permitía realizar las 100 corridas en cada uno, sin

embargo fue también importante realizar experimentos con el problema grande ya que fue donde principalmente se encontraron diferencias entre los algoritmos, además que los problemas reales son largos y el algoritmo debe poder resolverlos satisfactoriamente.

Como el algoritmo genético versión 3, de tiempo de ciclo variable para cada individuo y función objetivo enfocada a la eficiencia de la línea, fue el que obtuvo mejores soluciones se utilizó para realizar pruebas adicionales con problemas *benchmark* obtenidos de [28]. Este libro del autor Armin Scholl, concentra una serie de problemas que han sido propuestos por diversos investigadores para probar los algoritmos de balanceo de líneas de ensamble que tienen como objetivo encontrar la mejor distribución de tareas cuando se tiene un número definido de estaciones. Los resultados se muestran en la tabla 4.8 mostrada a continuación. Como podemos ver en las últimas dos columnas se encuentran por un lado el tiempo de ciclo que debía salir al final, y en la última columna el tiempo de ciclo que logró el algoritmo genético con tiempo de ciclo variable. Prácticamente en todos los casos se llega al tiempo de ciclo que debía obtener. Una observación importante respecto a estas pruebas realizadas, es que en cada corrida se llegó consistentemente al Tc mostrado, por lo cual estos resultados no tienen desviación estándar.

Cuadro 4.8: Resultados de tiempo de ciclo de problemas *benchmark* obtenidos del algoritmo genético con tiempo de ciclo variable

Instancia	Número de estaciones	Tc *Benchmark*	Tc logrado por AG con Tc var
1	8	1860	1860
2	9	1638	1638
3	10	1526	1526
4	11	1400	1412
5	12	1400	1400

4.3.5. Ejemplos de Soluciones Obtenidas

Para verificar que tan similares son las mejores soluciones obtenidas con los algoritmos y la solución óptima, se tomaron algunos ejemplos, para el problema 1 se muestran a continuación estos ejemplos.

En el cuadro 4.9 tenemos tres soluciones con sus respectivos valores del nivel de balance de cada una, también en el primer renglón se encuentra la distribución óptima a la cual se debería haber llegado, aunque todas las soluciones son diferentes con res-

Cuadro 4.9: Mejores soluciones obtenidas utilizando el problema 1

	Balance	Distribución tareas 1 a 18																
Óptimo	100.00	5	2	6	4	1	8	1	1	2	7	7	1	6	2	6	5	4	7
Solución 1	96.15	5	1	6	6	1	5	1	3	3	6	7	1	6	2	7	4	5	7
Solución 2	98.71	5	3	7	6	1	5	1	2	3	7	7	1	5	2	4	3	4	6
Solución 3	99.01	4	2	7	7	1	7	1	1	2	4	7	1	5	2	8	3	3	5

	Distribución tareas 19 a 35																Dif	% Dif
Óptimo	3	3	3	6	4	3	8	4	6	5	5	8	7	2	7	8	5	-	-
Solución 1	4	2	3	7	4	2	8	3	6	4	8	5	8	2	7	8	8	19	54.29
Solución 2	5	2	3	7	4	2	1	4	6	4	3	6	7	1	7	6	5	20	57.14
Solución 3	5	3	4	7	4	3	6	6	6	4	8	5	8	2	8	6	8	22	62.86

pecto al óptimo, se puede verificar que aunque no exista similitud, el nivel de balanceo obtenido no es del todo malo, por el contrario, el obtener soluciones mas parecidas a la óptima no nos asegura obtener también un buen nivel de balanceo.

De la misma manera se revisaron los resultados obtenidos utilizando el problema 2 y la conclusión a la que se llegó fue la misma. En el siguiente cuadro 4.10 se resumieron las soluciones, y debido a que el cromosoma es largo no se incluye el detalle del mismo.

Cuadro 4.10: Mejores soluciones obtenidas utilizando el problema 2

	Balance	Diferentes	% Diferentes
Solución 1	92.11	72	72.00
Solución 2	92.31	71	71.00
Solución 3	98.12	77	77.00

4.4. Resumen

En este capítulo se presentaron los experimentos y resultados preliminares de esta investigación. Después de haber realizado estos experimentos y analizado los resultados de cada uno de ellos concluimos en continuar el desarrollo de nuestro modelo mediante el uso de la versión 3 del algoritmo genético con tiempo de ciclo variable, ya que ofrece mejores resultados a la solución del problema en términos de la eficiencia obtenida y el tiempo de ejecución. De las conclusiones principales obtenidas de estos experimentos es que a pesar de obtener buenos resultados de balanceo, la población convergió antes de haber logrado el balanceo óptimo para el caso de los problemas obtenidos del generador de problemas, aunque por otro lado si logró en la mayoría de los casos los resultados deseados de los problemas *benchmark*. En adelante se tratará de mejorar estos resultados de balanceo e incrementar la dificultad del problema añadiendo restricciones adicionales.

Capítulo 5

Encadenamiento de Heurísticas para mejorar el Balanceo de Líneas de Ensamble de Modelo Mixto

Este capítulo presenta los experimentos y resultados finales derivados de esta investigación. Hasta este momento solamente se tenían las tres versiones del algoritmo genético presentados en el capítulo anterior y después de llevar a cabo los experimentos anteriores decidimos continuar desarrollando el algoritmo genético con tiempo de ciclo variable por individuo. Sin embargo, este procedimiento prácticamente solo respeta restricciones de precedencia, ahora es tiempo de incluir otro tipo de restricciones. Se presentan los resultados obtenidos de tres nuevos métodos descritos en la siguiente sección.

5.1. Algoritmos Implementados

Para continuar desarrollando un método que nos permitiera el manejo de restricciones suaves o preferencias del usuario, se implementaron y probaron las tres soluciones siguientes.

5.1.1. Algoritmo: Balanceo con Heurísticas - Algoritmo Genético - Min Conflicts (BH-AG-MC)

Este algoritmo está compuesto por tres etapas: la etapa heurística, un algoritmo genético y un algoritmo que implementa la heurística min-conflicts. Las dos primeras etapas permanecen igual a la del algoritmo genético con tiempo de ciclo variable del capítulo anterior, el cual nos había dado los mejores resultados de balanceo. La misma etapa de balanceo heurístico es utilizada también por las otras dos soluciones implementadas para los experimentos finales. Esta etapa está descrita en el capítulo 3 en el algoritmo 5. La etapa del Algoritmo Genético se describió en el algoritmo 13. El paso que se ha agregado utiliza un algoritmo basado en la técnica de búsqueda local Min-Conflicts, y se describió en el capítulo 3 en el algoritmo a continuación en el algoritmo 10. Para calcular la aptitud en esta etapa del algoritmo min-conflicts también se utiliza

la función objetivo definida en la etapa del algoritmo genético.

5.1.2. Algoritmo: Balanceo con Heurísticas - Recocido Simulado - Min Conflicts (BH-RS-MC)

Este método consiste en tres etapas también, utiliza la etapa de balanceo con heurísticas ya descrita en secciones anteriores, una etapa basada en el algoritmo de Recocido Simulado descrita a continuación en el algoritmo 14 y finalmente una tercera etapa utilizando el algoritmo descrito en 10.

5.1.3. Algoritmo: Balanceo con Heurísticas - Algoritmo Memético - Min-Conflicts (BH-AM-MC)

En este método también se tienen tres etapas, en la primera etapa generamos la población del algoritmo memético. En la segunda etapa, utilizamos un algoritmo genético, sin embargo ahora adicionamos una variante que es el uso del algoritmo de recocido simulado para realizar la mutación, este algoritmo memético se describió en el algoritmo 7. Y la etapa Min-Conflicts permanece sin cambios, tal como se describió en el algoritmo 10. El operador de mutación que utiliza el recocido simulado está descrito también en el capítulo 3, en el algoritmo 9.

5.2. Descripción de los Problemas

Los problemas utilizados fueron obtenidos del generador de problemas. Se obtuvieron problemas con diferentes parámetros. Los parámetros que se variaron fueron principalmente, el número de estaciones, el número de tareas, y el tiempo de ciclo, ya que éstos afectan en mayor medida el tiempo requerido por los algoritmos para solucionarlos. En cuanto al número de estaciones se comenzó con 8 estaciones ya que una instancia con menor número no reflejaría un problema real. Se terminó con un máximo de 50 estaciones. Es difícil determinar un número máximo de estaciones permitido, ya que entre más larga sea una línea de ensamble en la práctica se vuelve menos confiable ya que se pudieran incrementar los paros de línea, esto es, disminuir el tiempo promedio entre fallas. Las características de los problemas que se utilizaron para llevar los experimentos finales se describen en el cuadro 5.1 de donde permanecen constantes el número de modelos $M = 2$ y el porcentaje de demanda por producto $q_m = [0.5, 0.5]$, en lo que respecta al resto de los parámetros fueron diferentes. Las columnas en la tabla representan:

k = Número máximo de estaciones u operadores permitidos

Algoritmo 14 Algoritmo BH-RS-MC: Etapa Recocido Simulado

1: **ETAPA Recocido Simulado**

1. Proporcionar temperatura inicial (Ti).
2. Proporcionar número de ciclos (Nc).
3. Mientras la temperatura no sea cero o el valor óptimo de balanceo no se haya encontrado realizar los siguientes pasos:

   - Durante 10 veces realizar:

     a) Elegir una tarea aleatoriamente.

     b) Verificar todos los movimientos posibles para la tarea elegida.

     c) Elegir uno de esos movimientos aleatoriamente para obtener el nuevo individuo.

     d) Para llevar a cabo la nueva asignación verificar si T = 0, devolver el individuo actual

     Si no, calcular Δ.
     Δ = B del nuevo individuo - B del individuo actual.
     Donde B es calculado con la ecuación 3.3

     Si $\Delta > 0$ entonces el individuo actual = nuevo individuo.

     Si no, con una probabilidad $p = exp(\frac{\Delta}{T})$ aceptar o no el nuevo individuo.

4. Con cada asignación guardar el último individuo y el mejor individuo encontrado.

n = Número de tareas

TO = Tiempo de ciclo óptimo

WR = West ratio

Rv = Radio de variabilidad de tiempo

C = Número de tareas que pudieran provocar conflicto de restricciones suaves

Cuadro 5.1: Características de los problemas utilizados en la experimentación

Instancia	n	k	TO	WR	Rv	C
1	35	8	100	4.375	10	21
2	97	10	100	9.7	48	20
3	101	15	130	6.73	59	55
4	107	20	150	5.35	95	40
5	194	10	100	19.4	21	20
6	195	15	150	13	38	30
7	201	20	200	10	68	40
8	405	20	150	20.25	26	38
9	402	40	150	10	49	42
10	740	50	150	14.8	90	97

Como se observa en los datos en general los problemas incrementan su complejidad. Para los primeros experimentos presentados a continuación utilizamos como base el primero y el último problema ya que se encuentran entre los más fáciles y los más difíciles, respectivamente.

5.3. Experimentos

A continuación se describen los experimentos finales realizados. Estos experimentos utilizan los problemas descritos en la tabla 5.1. En la primera parte describiremos un experimento realizado para afinar uno de los parámetros en la etapa Min-Conflicts y posteriormente se utilizan los tres algoritmos implementados con cada una de las instancias para verificar el comportamiento de los mismos y decidir en base a ellos el algoritmo que tomaremos como el modelo de solución.

5.3.1. Experimentos con algoritmo BH-AG-MC

Para desarrollar este procedimiento, se utilizó como base la técnica Min-Conflicts, sin embargo, debido a que al aplicarla su algoritmo básico, no devolvía buenos resultados, se comenzaron a modificar ciertos aspectos y para llegar a dicha solución, se realizaron algunos experimentos. En este caso utilizamos el algoritmo BH-AG-MC que es una extensión de la versión 3 del algoritmo genético descrito en el capítulo anterior, excepto que ahora se incluye la etapa Min-Conflicts como se describe en el algoritmo 10. A continuación se presentan los experimentos que fueron realizados.

Afinación Etapa Min-Conflicts

En este algoritmo se especifica un porcentaje que permite que aunque las restricciones suaves mejoran la aptitud se puede empeorar en un 10 %, para utilizar ese valor se realizaron pruebas con valores más grandes. Los resultados se muestran en los cuadros 5.2 y 5.3, para el problema 1 y 10, respectivamente. Este experimento fue hecho durante 15 veces en cada caso, por lo que las tablas muestran resultados promedios obtenidos así como su desviación estándar.

Cuadro 5.2: Resultados de afinación de la etapa Min-Conflicts para un problema corto

Margen a empeorar	Etapa BH		Etapa AG		Etapa MC	
permitido	Balance	Conflictos	Balance	Conflictos	Balance	Conflictos
25 %	84.01	4.17	98.77	4.5	86.1	3.33
σ	0.01	0.38	0.053	0.522	0.10	1.15
20 %	84.17	4.17	98.71	4.92	78.82	2.67
σ	0.03	0.57	0.065	0.90	0.091	0.88
10 %	85.44	4.25	98.57	4.42	95.78	4.08
σ	0.024	0.45	0.005	0.90	0.05	1.08

Observamos que aunque en el problema largo no se ve tanta diferencia entre los resultados si existe con el problema 1, en general se comporta mejor cuando le dejamos como parámetro un 10 % como margen para empeorar. En los cuadros, la primera columna nos dice el margen que se le está permitiendo al algoritmo que empeore la aptitud en la etapa Min-Conflicts; la columna Balance, se refiere al valor de la eficiencia o nivel de balance obtenido en cada etapa; y la columna Conflictos, nos indica el número de estaciones que presentan conflicto debido a las restricciones suaves.

Cuadro 5.3: Resultados de afinación de la etapa Min-Conflicts para un problema largo

Margen a empeorar	Etapa BH		Etapa AG		Etapa MC	
permitido	Balance	Conflictos	Balance	Conflictos	Balance	Conflictos
25 %	81.89	16.3	91.40	16.7	87.29	9
σ	0.0046	2.26	0.0123	2.213	0.029	1.56
20 %	82.10	16.8	90.19	14.2	84.08	7.4
σ	0.005	2.16	0.004	0.833	0.04	1.14
10 %	81.92	16.4	91.18	17.1	87.37	9.6
σ	0.0033	1.57	0.0143	2.64	0.023	1.64

Función objetivo

Ahora bien, en ese procedimiento que probamos existe un detalle, la etapa Heurística se dedica a la generación de la población para el algoritmo genético, posteriormente el algoritmo genético trata de maximizar la eficiencia sin importarle las restricciones suaves y finalmente la etapa Min-Conflicts trata de mejorar las restricciones suaves permitiendo empeorar la eficiencia, entonces como resultado de esto durante las últimas dos etapas nos olvidamos de uno u otro objetivo lo que nos lleva a obtener una no muy buena solución. Para mejorar esto se propone pasar de la función objetivo de la ecuación 4.9 a la función 3.2, la cual será utilizada por ambas etapas. Esta nueva función objetivo combina tanto la eficiencia como el nivel de cumplimiento de las restricciones suaves.

Los resultados de la comparación de funciones objetivo se muestran en los cuadros 5.4 y 5.5 para el problema 1 y 10, respectivamente. Para que se pudieran comparar estos resultados el valor mostrado en la columna Balance proviene de la ecuación 4.9 en todos los casos, aunque en los algoritmos fue utilizada una u otra función según corresponda.

Los resultados de este experimento muestran que con la nueva función objetivo mejoró en ambos casos y aunque en el problema 1 (el corto), durante la etapa de Min-Conflicts ya no mejoró si lo hizo en el problema mas grande.

A continuación se muestra un ejemplo de cómo evoluciona la solución al problema 10 con el uso de gráficas del resultado obtenido mediante el modelo de solución. Este ejemplo utiliza el problema número 10 en el que se tienen como máximo 50 estaciones. La gráfica 5.1 contiene los resultados obtenidos al final de la etapa BH. Por su parte la gráfica 5.2 contiene los resultados obtenidos al final de la etapa del algoritmo genético. Por último tenemos la gráfica 5.3 que muestra resultados al final de la etapa Min-Conflicts, aunque visualmente no se nota gran diferencia, hay que recordar que la

Cuadro 5.4: Comparativa de funciones objetivo para un problema corto

Función	Etapa BH		Etapa GA		Etapa MC	
objetivo	Balance	Conflictos	Balance	Conflictos	Balance	Conflictos
4.9	85.44	4.25	98.57	4.42	95.78	4.08
σ	0.024	0.45	0.005	0.90	0.05	1.08
3.2	93.26	3.4	98.02	2	98.02	2
σ	0.018	0.38	0.005	0.90	0.35	1.16

Cuadro 5.5: Comparativa de funciones objetivo para un problema largo

Función	Etapa BH		Etapa GA		Etapa MC	
objetivo	Balance	Conflictos	Balance	Conflictos	Balance	Conflictos
4.7	81.92	16.4	91.18	17.1	87.37	9.6
σ	0.0033	1.57	0.0143	2.64	0.023	1.64
3.3	90.12	13.8	94.8	6.8	94.59	4.6
σ	0.005	1.92	0.007	2.05	0.009	1.22

mejora no solo depende del balanceo sino también de los conflictos en las estaciones debidos a restricciones suaves.

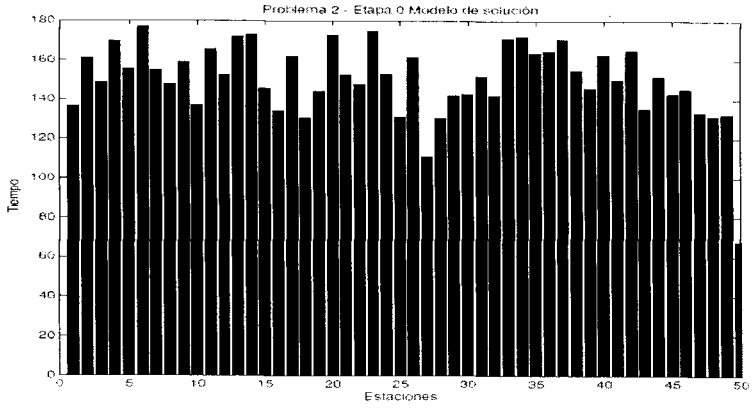

Figura 5.1: Tiempo de estaciones de un problema largo - Etapa BH

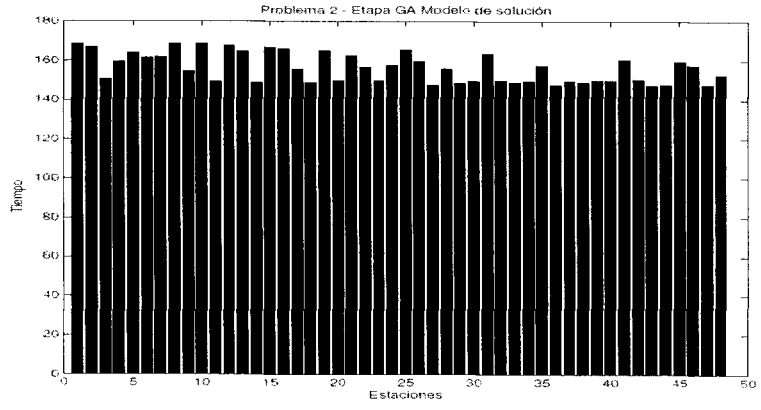

Figura 5.2: Tiempo de estaciones de un problema largo - Etapa AG

5.3.2. Comparativa General de los Algoritmos

A continuación, en el cuadro 5.6, se presentan los resultados obtenidos para cada instancia con cada uno de los algoritmos implementados para realizar la experimentación. Esta tabla muestra resultados promedios de las corridas realiadas y la desviación estándar en cada caso. La columna t(min) muestra el tiempo de ejecución del algoritmo en minutos. Balance se refiere a la eficiencia obtenida de acuerdo a la función objetivo de los algoritmos. Finalmente Tc muestra el tiempo de ciclo promedio alcanzado por las soluciones.

La primera columna dice a que instancia pertenecen los resultados mostrados en las siguientes columnas. Las primeras tres columnas posteriores a la instancia muestran

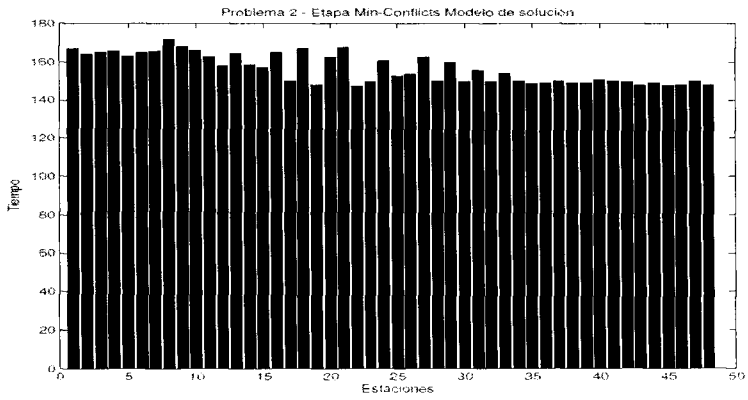

Figura 5.3: Tiempo de estaciones de un problema largo - Etapa MC

los resultados del algoritmo BH-AG-MC de tres etapas que consiste en balanceo con heurísticas, Algoritmo Genético y Min-Conflicts; las siguientes tres columnas contienen los resultados del algoritmo BH-RS-MC de tres etapas, balanceo con heurísticas, Recocido Simulado y Min-Conflicts. Finalmente, las últimas tres columnas incluyen los resultados del algoritmo BH-AM-MC por etapas que consiste en, balanceo con heurísticas, Algoritmo Memético y Min-Conflicts. En general, cada instancia fue probada por 10 corridas en cada algoritmo, a excepción de la última instancia que sólo fue probada durante 5 corridas para obtener los resultados mostrados. En el caso del algoritmo BH-AM-MC fue necesario antes de estas corridas realizar pruebas adicionales para ajustar los parámetros de temperatura inicial, número de generaciones y probabilidades de cruce.

Para mostrar estos resultados de manera gráfica se tiene la figura 5.4 el eje x representa cada instancia que fue probada con los algoritmos y el eje y representa el balanceo obtenido en cada instancia para el algoritmo correspondiente. La línea verde representa al algoritmo que contiene la solución que utiliza al algoritmo memético que como se había dicho antes es el que obtiene mejores resultados en la mayoría de los casos.

Las etapas intermedias son las que contribuyen a la diferencia de los diferentes encadenamientos presentados, ya que la primera y última etapas permanecen igual en las tres opciones de encadenamiento descritas. Para observar el comportamiento de la función objetivo a través de estas etapas intermedias se presentan a continuación las gráficas de mejor encontrado del mismo problema en los tres casos. La figura 5.5 muestra la evolución de la función objetivo en la etapa del algoritmo genético utilizando el algoritmo BH-AG-MC. El eje x representa el número de evaluaciones realizadas a través de las generaciones y el eje y muestra el nivel de balanceo alcanzado en estas evaluaciones. En cada generación se realizan 50 evaluaciones.

La figura 5.6 muestra la evolución de la función objetivo en la etapa del recoci-

Cuadro 5.6: Comparativa del nivel de balanceo de diferentes problemas obtenidos mediante el uso de los algoritmos implementados

Instancia	Algoritmo BH-AG-MC			Algoritmo BH-RS-MC			Algoritmo BH-AM-MC		
	t(min)	Balance	Tc	t(min)	Balance	Tc	t(min)	Balance	Tc
1	2	93.4	104	5	92.4	106	26	92.9	104
σ	0.35	0.01	3.27	1.73	0.007	2.16	0.01	0.008	0.76
2	7	94.9	105	5	94.2	112	7	96.6	101
σ	0.19	0.07	0.81	0.42	0.012	5.47	0.06	0.008	0.89
3	7	84.2	144	4	78.1	150	23	85.1	143
σ	0.04	0.014	6.19	0.025	0.008	1.52	0.14	0.011	4.14
4	7	93.3	165	4	88.7	177	25	95.2	159
σ	0.27	0.03	2.79	0.032	0.009	0.51	0.38	0.007	0.707
5	15	97.1	103	14	96.7	103	14	97.7	102
σ	2.13	0.008	0.23	2.08	0.014	1.71	1.03	0.002	0.353
6	15	93.9	163	13	93.5	180	12	97.11	158
σ	0.576	0.01	8.73	1.48	0.01	20.9	0.63	0.009	1.06
7	14	95.1	215	10	95.6	214	17	95.7	212
σ	1.55	0.03	0	4.7	0.001	13.8	1.65	0.005	1.59
8	41	93.7	168	16	92.8	194	40	94.8	162
σ	0.12	0.018	7.04	3.4	0.01	49.1	3.4	0.014	3.5
9	36	95.6	162	21	94.8	185	45	95.06	166
σ	12.4	0.013	4.23	3.73	0.006	29.38	7.64	0.004	7.7
10	130	93.7	172	78	97.73	169	185	96.5	171
σ	10.12	0.003	2	0.42	0.004	12.92	8.29	0.015	20.1

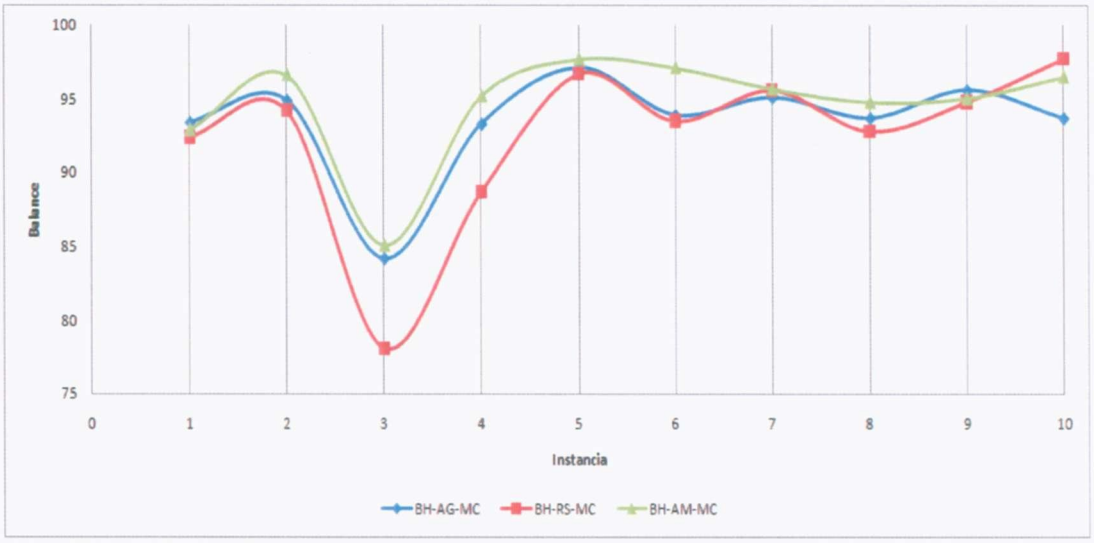

Figura 5.4: Comparativa gráfica del nivel de balanceo de diferentes problemas obtenidos mediante el uso de los algoritmos implementados

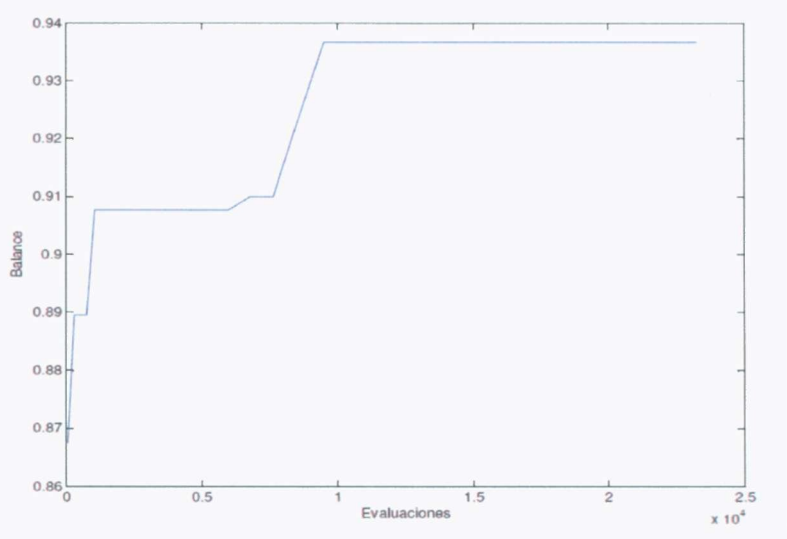

Figura 5.5: Ejemplo de gráfica de mejor encontrado del algoritmo BH-AG-MC en la etapa del algoritmo genético

do simulado del algoritmo BH-RS-MC. El eje x representa el número de iteraciones realizadas y el eje y muestra el nivel de balanceo alcanzado en estas evaluaciones.

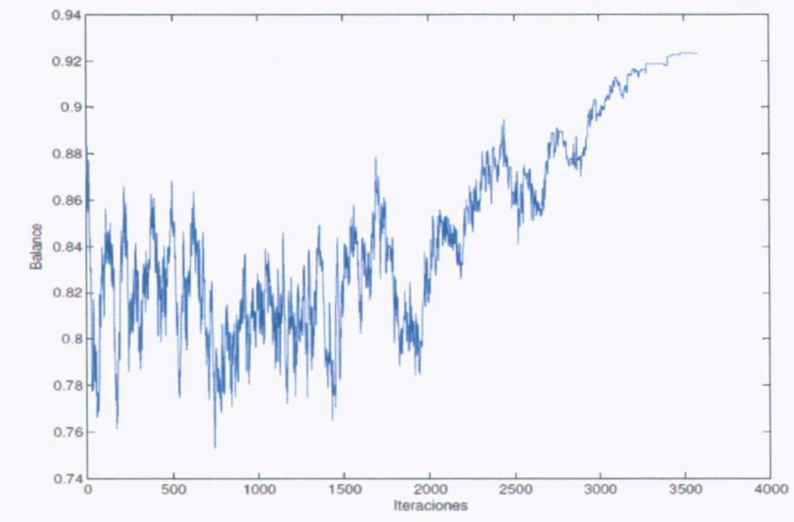

Figura 5.6: Ejemplo de gráfica de evolución del balanceo del algoritmo BH-RS-MC en la etapa del recocido simulado

La figura 5.7 muestra la evolución de la función objetivo en la etapa del algoritmo memético utilizando el algoritmo BH-AM-MC. El eje x representa el número de evaluaciones realizadas a través de las generaciones, en cada generación se realizan 50 evaluaciones y el eje y muestra el nivel de balanceo alcanzado en estas evaluaciones.

5.4. Resumen

En este capítulo se presentaron los experimentos y resultados que permiten demostrar las fortalezas y debilidades del modelo utilizado. El algoritmo que utiliza la técnica del Algoritmo Memético y el que incluye el Algoritmo Genético producen mejores resultados que el que no utiliza una técnica evolutiva. Entre estos dos se observan en más casos mejores resultados por parte del encadenamiento que utiliza al algoritmo memético. En cuanto al tiempo de ejecución obtuvo un tiempo mayor pero aún aceptable, esto debido a que tiene al recocido simulado como un procedimiento de mutación. El recocido simulado por utilizar una solución a la vez es el que toma menos tiempo y el algoritmo genético tiene un tiempo intermedio, aunque en algunos casos similar al memético. En cuanto al tiempo de ciclo obtenido al final de los algoritmos, también resultó mejor en la mayoría de las instancias que elegimos como el modelo de solución. Cabe mencionar que este algoritmo al ser más complejo requiere más experimentos de afinación, ya que son varios parámetros que en la presente investigación dejamos fijos.

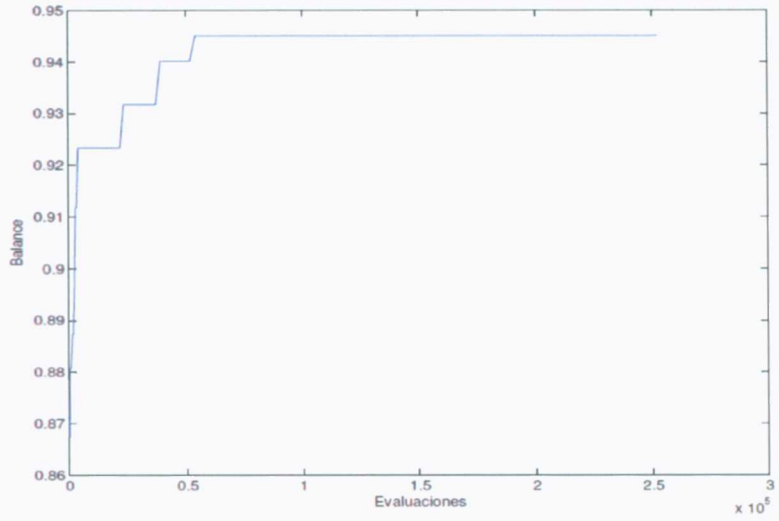

Figura 5.7: Ejemplo de gráfica de mejor encontrado del algoritmo BH-AM-MC en la etapa del algoritmo memético

También es importante notar que la última etapa es de gran ayuda en problemas más largos y complejos. El enfoque elegido, en la segunda etapa se concentra principalmente en la eficiencia de la línea y en la última etapa se concentra en las restricciones suaves sin descuidar el balanceo.

Capítulo 6

Conclusiones y Trabajo Futuro

En este capítulo se presentan las conclusiones y contribuciones derivadas de esta investigación, así como también algunas sugerencias para extender el trabajo presentado en este documento.

6.1. Conclusiones

La investigación realizada en esta tesis se enfocó en la solución al problema de balanceo de líneas de ensamble de modelos mixtos, demostrando que el uso de técnicas de inteligencia computacional tales como los Algoritmos Evolutivos y el Recocido Simulado son métodos que pueden ser aplicados para la solución de dicho problema ya que obtienen resultados competitivos como se observó con las instancias *benchmark* probadas. Además obtienen mejores resultados que las técnicas clásicas ya que éstas fueron utilizadas en la etapa de balanceo con heurísticas, y en todos los casos hubo mejora de una etapa a la siguiente.

Al principio del documento se especificó el planteamiento de los problemas a resolver, posteriormente se comenzó con el desarrollo del algoritmo, llevándose a cabo diversos experimentos hasta llegar al modelo de solución planteado en el capítulo 3. En el modelo de solución se plantearon el uso de dos algoritmos, un algoritmo memético y una técnica de satisfacción de restricciones, min-conflicts. Los resultados obtenidos eran esperados ya que teóricamente el algoritmo memético contiene una ayuda adicional en la parte de mutación que le ayuda a encontrar mejores soluciones y ambos algoritmos evolutivos al manejar múltiples soluciones tienen mayor probabilidad de obtener mejores soluciones que el recocido simulado que solo maneja una solución a la vez. En cuanto al Min-Conflicts, éste no compite con estos algoritmos ya que no es un algoritmo de optimización sino de satisfacción de restricciones.

Como se había mencionado antes, para la primera etapa del algoritmo se tomó como base el trabajo de Vilarinho y Simaria para generar la población inicial del Algoritmo Genético, ya que ayudaba a generar soluciones diferentes al combinar varias heurísticas Es importante mencionar que inclusive en esta etapa, los resultados de balanceo ya

son buenos, y las soluciones son factibles. De haberlos generados de manera aleatoria los resultados serían peores sin mencionar que existirían soluciones no factibles, ya que pudieran violar alguna restricción de precedencia.

En general, los resultados de cada experimento fueron conduciendo el desarrollo del modelo. Como se observó, en los cuadros de resultados, las soluciones encontradas en cada caso sobrepasan el 94 % de eficiencia de la línea, mejores resultados se obtienen entre más fácil sea el problema. Estos valores de eficiencia encontrados son bastante aceptables ya que nos indican que prácticamente un 6 % del tiempo el operador está ocioso. Si lo trasladamos en términos de tiempo, por cada hora trabajada el operador estaría inactivo aproximadamente 4 minutos, lo cual no es mucho tiempo considerando que típicamente se aceptan valores a partir del 80 % de eficiencia.

6.2. Contribuciones

Esta investigación contribuye a enriquecer el conocimiento acerca del uso de técnicas de inteligencia computacional sobre los problemas de balanceo de líneas de ensamble de modelos mixtos. De las principales contribuciones son los algoritmos desarrollados en esta tesis. Pudimos ver que sobre todo el último algoritmo desarrollado es bastante funcional, además se observó que no es necesario el uso de una función objetivo muy elaborada para llegar a buenos resultados, comparables con aquellas obtenidas por otros métodos. Se verificó también la efectividad de separar el manejo de las restricciones de precedencia del resto de las restricciones consideradas. Finalmente, se desarrollaron variaciones de los algoritmos de Recocido Simulado, Min-Conflicts y Algoritmos Genéticos para adaptarlos a las necesidades del problema de estudio. Cabe mencionar que el modelo de solución desarrollado puede ser implementado a problemas en la vida real de una forma relativamente simple. Como los algoritmos en general son sensibles a los distintos tipos de problemas, podrían requerirse ajustes a los parámetros. Debido a que se utilizó un algoritmo memético que incluye en su operador de mutación un algoritmo basado en la técnica del recocido simulado, los parámetros que requieren ajustes son: la temperatura inicial, la cantidad de ciclos internos y externos o probabilidades de mutación y cruce.

Con base en los resultados obtenidos es posible concluir que el modelo basado en la combinación de un Algoritmo Memético con Min-Conflicts es un excelente enfoque para la solución de estos problemas de balanceo.

6.3. Trabajo Futuro

A continuación se presentan algunas ideas sobre las cuales se podría extender el trabajo realizado en esta investigación. En cuanto al planteamiento del problema:

- Generar la secuenciación de los modelos, ya que se conocen el tiempo de ciclo requerido y la mezcla de los modelos a producir, el siguiente paso es la secuenciación de los mismos que nos dice en que orden deben ser producidos, ya que muchas veces cuando cambia la demanda de cada modelo también se ve afectada la secuenciación y por lo tanto el tiempo de ciclo. Entonces es indispensable encontrar una secuencia de producción que de cambiar la mezcla no dispare tanto el tiempo de ciclo de la línea.

- Implementar conceptos de valor agregado (VA) y no valor agregado (NVA) para buscar un algoritmo que identifique aquellas operaciones que no estén generando valor al producto y trate de minimizarlas o eliminarlas.

- Incluir conceptos de Bin Packing, actualmente utilizados en la industria automotriz. Mediante estas técnicas es posible reducir el número de estaciones requeridas, sin embargo, se debe poner especial énfasis en los materiales utilizados en cada estación.

En cuanto a las técnicas y el modelo de solución:

- Buscar nuevas heurísticas que ayuden a la generación de la población inicial del Algoritmo Genético.

- Definir otro esquema de codificación de las soluciones.

- Implementar otro algoritmo, que calibre automáticamente los parámetros para cada etapa del algoritmo, ya que al ser una gran cantidad de ellos, muchos los dejamos fijos durante la experimentación manual. Sin embargo, valdría la pena considerar este aspecto importante que incrementaría la viabilidad de su uso.

Bibliografía

[1] Christian Becker and Armin Scholl. A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 127(3):694–715, February 2006.

[2] T.O. Boucher. Choice of assembly line design under task learning. *International Journal of Production Research*, 1987.

[3] B. Bryton. Balancing of a continuous production line. Master's thesis, Northwestern University, 1954.

[4] Joseph Bukchin, Ezey M. Dar-El, and Jacob Rubinovitz. Mixed model assembly line design in a make-to-order environment. *Computers and Industrial Engineering*, 41(4):405–421, 2002.

[5] Yossi Bukchin and Ithai Rabinowitch. A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research*, 127(1):492–508, October 2006.

[6] Cintia Copaceanu. Mixed-model assembly line balancing problem: variants and solving techniques. In *Proceedings of ICMI*, pages 337–360, 2006.

[7] E.M. Dar-El. Malb-a heuristic technique for balancing large single-model assembly lines. *AIIE Transactions*, 1973.

[8] Goldman-R. Feldman, R.G. and W.M. Keyserling. Peripheral nerve entrapment syndromes and ergonomic factors. *American Journal of Industrial Medicine*, 1983.

[9] Alonso Perez S. Gerardo Sánchez Schmitz, Mario Barceló V. Sistema de apoyo a la toma de decisiones en el balanceo de líneas de manufactura flexible con un enfoque de gestión del conocimiento. In *Memorias del II Simposio Internacional de Sistemas de Información e Ingeniería de Software en la Sociedad del Conocimiento (SISOFT 2003)*.

[10] F Glover and M Laguna. Tabu search. *Journal of the Operational Research Society*, 1997.

[11] Birnie D. P. Helgeson, W. P. Assembly line balancing using the ranked positional weight technique. *Journal of Industrial Engineering*, 1961.

[12] Peker Ahmet Kara Yakup, Ozcan Ugur. Balancing and sequencing mixed-model just-in-time u-lines with multiple objectives. In *Applied mathematics and computation*, 2006.

[13] Wester L. Kilbridge, M. D. A heuristic method for assembly line balancing. *Journal of Industrial Engineering*, 1961.

[14] Ruiz A.B. Lapierre, S.D. Balancing assembly lines: An industrial case study. *Journal of the Operational Research Society*, 2004.

[15] Vilarinho P. M. A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 40:1405–1420(16), 15 April 2002.

[16] Vilarinho P. M. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type 2. *Computers Industrial Engineering*, 47, 2004.

[17] Vilarinho P. M. 2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers and Industrial Engineering*, 2007.

[18] Patrick R. McMullen and Gregory V. Frazier. A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. *International Journal of Production Economics*, 51(3):177–190, September 1997.

[19] Patrick R. McMullen and Peter Tarasewich. Using ant techniques to solve the assembly line balancing problem. *IIE Transactions*, 35(7):607–617, 2003.

[20] Young H.H. Moodie, C.L. A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *Journal of Industrial Engineering*, 1965.

[21] Armin Scholl Nils Boysen, Malte Fliedner. A classification of assembly line balancing problems. *European Journal of Operational Research.*, 2007.

[22] Armin Scholl Nils Boysen, Malte Fliedner. Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 2008.

[23] Stephen Pheasant. *Bodyspace: Anthropometry, Ergonomics and Design.* Routledge, UK, 2 edition, 1996.

[24] Ye Lu Ping Su. Combining genetic algorithm and simulation for the mixed-model assembly line balancing problem. 2007.

[25] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 2003.

[26] P. Aravindan S. G. Ponnambalam and G. Mogileeswar Naidu. A multi-objective genetic algorithm for solving assembly line balancing problem. In *International journal, advanced manufacturing technology*, 2000.

[27] M. E. Salveson. The assembly line balancing problem. *Journal of Industrial Engineering*, (6):18–25, 1955.

[28] Armin Scholl. *Balancing and Sequencing of Assembly Lines.* Springer-Verlag New York Inc, 1999.

[29] Fine L.J. Armstrong T.A. Silverstein, B.A. Occupational factors and carpal tunnel syndrome. *American Journal of Industrial Medicine*, 1987.

[30] Hua-Ming Song. Co-optimization of level schedules for mixed-model assembly line in just-in-time production system. *Machine Learning and Cybernetics*, 1:96–101, Aug. 2005.

[31] Silverstein BA Leonard J. Stetson D, Keyserling WM. Observational analysis of the hand and wrist: A pilot study. *Applied Occupational Environmental Hygiene.*, 1991.