Planificación de la vulcanización de neumáticos mediante modelos de optimización

Informe Final de Proyecto de Grado

Ingeniería de Computación

Joaquín Velázquez

Tutores:
Héctor Cancela
Pedro Piñeyro

Instituto de Computación
Facultad de Ingeniería
Universidad de la República
Uruguay
23 de agosto de 2018

Resumen

La fabricación de neumáticos se compone de una secuencia de etapas, entre las cuales se encuentra la de vulcanizado, etapa en la cual se la da su forma y consistencia final. Este proceso se caracteriza por ser costoso en términos de consumo energético y de tiempo, ya que además de los tiempos propios del proceso, existen tiempos de configuración que dependen del orden en que se realice el procesamiento de los distintos tipos de neumáticos. Estos están relacionados a la colocación de piezas en los moldes y de moldes en las máquinas de vulcanizado llamadas heaters.

En este proyecto de grado se estudió el problema de planificación de vulcanizado de neumáticos, en el marco de una actividad de extensión con la cooperativa uruguaya FunsaCoop. El objetivo consistió en desarrollar una herramienta informática que permita el manejo eficiente de los datos necesarios para la definición del problema y la aplicación de procedimientos de optimización para encontrar soluciones que minimicen el tiempo de vulcanizado, considerando los requerimientos de demanda para cada tipo de neumático. Para lograr el objetivo planteado, se elaboró en primera instancia una formulación de programación matemática, con el fin de obtener un plan de vulcanizado de duración mínima, teniendo en cuenta la demanda de cada neumático, los tiempos de configuración, la compatibilidad entre moldes y heaters, y la cantidad de piezas disponibles. Para resolver el problema se propusieron y evaluaron tres procedimientos de resolución, basados en técnicas exactas, heurísticas e híbridas, respectivamente. El procedimiento exacto consistió en resolver la formulación matemática propuesta mediante el software CPLEX. Para instancias grandes este procedimiento no resultó adecuado debido al tiempo de cómputo requerido. El segundo procedimiento de resolución propuesto está inspirado en el funcionamiento de la meta-heurística GRASP, mediante el desarrollo de heurísticas deterministas y aleatorizadas. Con las heurísticas fue posible resolver de forma óptima el 81 % de las pruebas realizadas en menos de 2 segundos de cómputo. Cabe destacar que, para instancias grandes, donde el software CPLEX no fue capaz de obtener soluciones factibles, las heurísticas encontraron soluciones en menos de 20 segundos. Por último, el tercer procedimiento de resolución propuesto, está basado en combinar los métodos heurísticos y exacto desarrollados. En este caso, las heurísticas se utilizan como un estimador del tiempo necesario para llevar a cabo la vulcanización. Una buena estimación del tiempo necesario para la vulcanización es de gran ayuda para resolver la formulación matemática propuesta, ya que la cantidad de variables y restricciones del modelo dependen en gran medida del valor del horizonte de planificación. Como resultado, se obtuvieron mejoras notorias en los tiempos de ejecución por parte de CPLEX y fue posible reducir los casos donde no se podían obtener soluciones óptimas. Comparando las soluciones obtenidas con las proporcionadas por FunsaCoop para casos reales, se puede concluir que el procedimiento híbrido es capaz de obtener soluciones de muy buena calidad en un tiempo razonable de cómputo.

Además de los procedimientos de resolución propiamente dichos, se desarrolló una aplicación web para facilitar la definición del problema, carga y edición de datos, ejecución de los procedimientos de resolución propuestos con distintas configuraciones y visualización adecuada de los resultados.

Palabras claves— Planificación de la Producción, Vulcanizado de Neumáticos, Lot-Sizing, Scheduling, Optimización, GRASP

Trabajos relacionados

El desarrollo de este proyecto de grado a dado lugar a las siguientes presentaciones en conferencias internacionales:

IFORS 2017

Título A tire curing scheduling problem

Autores Héctor Cancela, Pedro Piñeyro, Sofía Lemes, Agustín Ghioldi, Joaquín Velázquez

Lugar Québec, Canadá

EURO/ALIO 2018

Título A MILP formulation for a tire curing scheduling problem

Autores Héctor Cancela, Pedro Piñeyro, Joaquín Velázquez

Lugar Bolonia, Italia

CLAIO 2018

Título Analysis of MILP formulations for a tire curing scheduling problem

Autores Hector Cancela, Pedro Piñeyro, Nelson Troncoso, Óscar C. Vásquez y Joaquín Velázquez

Lugar Lima, Perú

Agradecimientos

En primer lugar, me gustaría dar las gracias a mis tutores Héctor Cancela y Pedro Piñeyro por toda su dedicación, ayuda y oportunidades que me han dado durante el desarrollo de mi investigación. Por sus correcciones y consejos durante la realización de toda esta tesis.

También me gustaría dar las gracias a Sofía Lemes y Agustín Ghioldi por su ayuda y aporte de su trabajo ya que fue el punto de partida de toda la tesis. A Óscar C. Vásquez por sus revisiones y correcciones cuando trabajamos en la formulación del problema. También quiero agradecer a Fernando Zabaleta, Enrique Romero y a sus compañeros de trabajo, quienes nos abrieron los puertas de FunsaCoop y dispusieron de su tiempo para explicarnos y brindarnos información de todo lo relevante para llevar a cabo este trabajo. También quiero agradecer a mis compañeros de trabajo quienes me han ofrecido consejos y flexibilidad en el transcurso de la carrera y la tesis.

Me gustaría dar las gracias a toda mi familia y amigos por el apoyo incondicional a lo largo de los años. Su ánimo y ayuda han sido muy importante para la realización de este trabajo.

¡Gracias!

Índice general

CAPÍTULO 1

Introducción

El objetivo de este documento es describir el trabajo de proyecto de grado sobre un caso particular de planificación de la producción. Específicamente se enfrenta al problema de planificación de la producción en el proceso de vulcanizado de neumáticos. La vulcanización es la fase encargada de darle la consistencia final al neumático, caracterizada por ser una de las más costosas del ciclo debido al consumo energético, desgaste de maquinaria y el tiempo implicado. La linea de producción en esta fase se compone por un grupo de máquinas con distintas especificaciones, capacidades y requerimiento de piezas, el cual pueden realizar tareas en simultaneo. Su uso coordinado es esencial para reducir costos y tiempos de producción. Pese a esto muchas organizaciones generan una planificación de forma manual y basada en la experiencia acumulada.

Este proyecto es una extensión del trabajo realizado por estudiantes del curso Optimización de Problemas de Producción, donde su tarea fue realizada en un marco de aplicación para la fábrica de neumáticos FunsaCoop. Esta cooperativa enfrenta el problema de ordenamiento de tareas en la etapa de vulcanizado, donde la planificación de la producción se hace de forma manual y basada en planificaciones realizadas anteriormente. El proceso de producción en FunsaCoop funciona en base a órdenes realizadas por clientes. Conociendo cuántos neumáticos se precisan producir la fábrica posteriormente se encarga de generar un inventario de neumáticos "crudos". Estos tipos de neumáticos no poseen una consistencia suficiente para ser un neumático propiamente. Posteriormente estos neumáticos crudos pasan a la etapa de vulcanizado. Esta etapa se encarga de darle la consistencia final a los neumáticos a través de las "prensas", también conocidos como "heaters". En la linea de producción se cuentan con varios tipos de heaters que difieren en los tipos de neumáticos que pueden producir y en capacidades. La salida de producción por la tanto está limitada en la capacidad de mantener la mayor cantidad de heaters produciendo. Por otro lado las instalaciones de FunsaCoop cuentan con una caldera a leña destinada a brindar el 80 % de su capacidad a los "heaters", que son las máquinas que realizan el proceso de vulcanizado. Dado que el consumo diario ronda en 25 a 30 toneladas de leña, incluyendo además las fechas de entrega y costos de mantenimiento es importante explorar procedimientos que aprovechen adecuadamente los recursos disponibles. Reducir el tiempo que la caldera está activa implica también disminuir la emisión de dióxido de carbono, el cual hoy en día afecta el calentamiento global entre otros factores medio ambientales. El problema a resolver consiste entonces en generar una planificación que indique los neumáticos que debe producir cada heater en periodos de tiempo bajo ciertas restricciones de capacidad. Tal planificación debe minimizar el tiempo de producción total y aprovechar el uso de recursos. Este tipo de problema está categorizado como problemas combinatorio, conocidos por su complejidad para resolverlos.

El objetivo de este proyecto es brindar una herramienta que permita gestionar los datos necesarios para la definición del problema y que incluya métodos de optimización, posiblemente heurísticos para encontrar soluciones que maximicen el desempeño del sistema. Para lograrlo en el transcurso del proyecto se pasó por varias etapas.

En una primera instancia se realizó un estudio de los problemas de ordenamiento, abarcando definiciones, características y clasificaciones. En particular se estudió los modelos Discrete Lot Sizing (DLS) y Job Shop Scheduling Problem (JSP) por sus aplicaciones en la planificación de la producción y similitudes con el problema a tratar en ese proyecto. Se analizaron además trabajos donde se aplica DLS, JSP y otros modelos para resolver problema de vulcanizados e industrias con características similares. Estos problemas suelen simplificarse para acotar la realidad tratada y facilitar la forma de resolverlo. Sin embargo pueden llegar a ser muy difícil de resolver debido a la cantidad de combinaciones y restricciones que pueden manejar. Por esto la planificación de la producción es considerado un problema combinatorio.

Existen varias formas de resolver este tipo de problemas y dependiendo de ella varían factores como el tiempo de resolución y calidad de solución. Por eso la siguiente etapa consistió en hacer un relevamiento de algoritmos y técnicas empleadas en la actualidad para resolver problemas combinatorios. La temática algoritmia es de propósito general, sin embargo fue posible crear una base teórica para aplicar procedimientos y prácticas habitualmente empleadas. Dentro de las técnicas aplicadas se pueden agrupar en técnicas exactas, de aproximación e híbridas. Cada una tiene sus ventajas y desventajas, el cual en este proyecto fueron exploradas. Entre los trabajos revisados, se incluyen además otros relacionados con el problema de vulcanizado e industrias similares para ver que tendencias hay respecto a las técnicas aplicadas en la actualidad.

Estas dos primeras partes se encuentran disponibles en los Anexos A y B. Se aconseja al lector su lectura previa para un mejor entendimiento de todo el documento.

Una vez elaborado el "Estado del Arte" se procedió a trabajar en la formulación matemática para modelar el problema de FunsaCoop. Como resultado en este proyecto se presenta una formulación MIP (Programación mixta-entera) el cual genera un ordenamiento de la producción que minimiza la cantidad de tiempo requerido para satisfacer la demanda de los clientes. Minimizar el tiempo total de producción implica también reducir el tiempo ocioso de la caldera a leña y maximizar el uso de recursos. La formulación matemática se elaboró gracias a la integración de varias actividades teóricas y prácticas. Una de ellas corresponde a las visitas realizadas en las instalaciones de FunsaCoop, que hizo posible el relevamiento de datos numéricos, forma de producir neumáticos, operativa de maquinaria y piezas, gerencia de recursos y demanda entre otros aspectos cruciales en la producción y planificación en la planta. Junto a otros artículos se vio los estándares de calidad que se aplican en la fabricación de neumáticos, como influye la calibración de la maquinaria utilizada y costos que conlleva su producción. Por otro lado se realizaron cursos dado por la Facultad de Ingeniería como forma de complementar puntos teóricos y aplicaciones prácticas en el ámbito de problemas combinatorios. Uno de ellos fue el curso "Optimización Combinatorial y Relajación Lagrageana" dado por el Dr. Francisco Barahona de IBM Research y reconocido en diversos temas de optimización combinatoria. El otro curso "Técnicas de Descomposición en Programación Matemática" dictado por el Dr. Víctor M. Albornoz de la Universidad Técnica Federico Santa María de Chile. Además se incluye la revisión de trabajos que enfrentan el problema de vulcanizado y que poseen características similares al ámbito de FunsaCoop, lo cual fueron útiles como referencia. Entre ellos se encuentran Degraeve and Schrage [1], Jans y Zeger [2], Shengping y Yang [3] y Degraeve [2] los cuales serán detallados más adelante.

Cabe destacar que la formulación elaborada omite y asume ciertos aspectos de la realidad, ya que de lo contrario la complejidad tratada sería aun mayor. Por ejemplo se asumió que todos los heater pueden producir a lo sumo dos neumáticos en simultaneo, que se cuenta con suficientes trabajadores para realizar tareas de configuración y mantenimiento, no se considera ubicación y traslado de piezas, y que la planificación no será modificada en ningún periodo de tiempo. Esto permitió simplificar la generación de un ordenamiento de la producción.

Parte del trabajo incluye pruebas de validación. Aquí se desarrollaron un conjunto de escenarios básicos que ayudaron a corroborar las soluciones dadas por el modelo. De esta forma los escenarios más complejos que surgen de combinar los sencillos se pudieron corroborar. Para realizar la validación se codificó la formulación matemática al lenguaje AMPL[1] (A Mathematical Programming Language), un lenguaje utilizado para representar modelos matemáticos. Para resolverlo se utilizó el software comercial CPLEX[2] ya que es capaz de interpretar la codificación AMPL.

Una característica de la formulación es que cuenta con un horizonte de planificación finito. Esto significa que se cuenta con un tiempo máximo para producir todos los neumáticos y la solución óptima debe ser menor. Este valor está determinado por el parámetro THB (Time horizon bound) e influye en la cantidad de variables y restricciones empleadas por el modelo. Si se emplea un valor alejado a la solución óptima la dimensión del modelo aumenta considerablemente ya que la cantidad de variables y restricciones implicadas aumentan. Por lo tanto debe ser estimado de forma cuidadosa.

La formulación matemática se puede resolver de varias maneras, muchas de ellas estudiadas en el Estado del Arte. En este proyecto se elaboró y evaluaron tres procedimientos para resolverla, donde en cada una se aplican técnicas exactas, heurísticas e híbridas respectivamente. Cada una de las formas busca una planificación de la producción óptima.

El procedimiento exacto consistió en resolver la formulación matemática por medio del software CPLEX. CPLEX es un software comercial conocido y desarrollado por IBM. Su potencial viene dado por la capacidad de analizar y aplicar varias técnicas exactas para encontrar una solución óptima. Adicionalmente se incluye un

[1] AMPL - https://ampl.com/
[2] CPLEX - https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer

estimador del parámetro THB ya que el modelo lo requiere. Esta estimación se hizo por medio de una ecuación que genera una cota superior de la solución óptima. Este estimador no es bueno en algunos escenarios dado que puede generar valores grandes. Como resultado se vio que CPLEX no es capaz de resolver los escenarios de tamaño mediano y grandes (correspondientes a casos reales) debido a la dimensiones del modelo. También se vio que los tiempos de ejecución llegaron a exceder las 8 horas debido a que se aplican técnicas exactas, el cual hacen un análisis exhaustivo del espacio de búsqueda. Además para usar CPLEX se requiere licencia comercial que conlleva un gasto extra y hasta posiblemente innecesario.

Para resolver varios de los problemas que surgieron se elaboró un segundo procedimiento que aplica técnicas heurísticas. Las heurísticas son rutinas simples y utilizadas para agilizar la exploración del espacio de búsqueda sin hacer un análisis exhaustivo. Esto permite reducir los tiempos de resolución notoriamente comparado con las técnicas exactas, sin embargo se sacrifica calidad de solución. Por eso estas técnicas se caracterizan por dar soluciones aproximadas a las óptimas y por su velocidad de convergencia. Para llevar a cabo este procedimiento en una primera instancia se elaboró un framework[3] inspirado en la meta-heurística GRASP.

Este framework resuelve problemas que puedan separarse en dos fases, una de construcción de soluciones y otra de mejora. La fase constructiva se forma por un conjunto de heurísticas constructivas encargadas de generar soluciones. La fase de mejora es un proceso opcional y destinado a mejorar o corregir las soluciones dadas en la fase constructiva. Esta herramienta se encarga de invocar a cada heurística constructiva de soluciones disponibles en la fase constructiva y en caso de contar con el proceso de mejora se aplica a cada solución generada. Además se emplean otras técnicas como la generación de soluciones en simultaneo para reducir aun más los tiempos de resolución. Según las heurísticas constructivas que se utilicen y como está desarrollada la fase de mejora se pueden obtener fácilmente otros algoritmos. Por ejemplo si se utiliza una heurística que construye aleatoriamente soluciones y la fase de mejora corresponde a una búsqueda local se obtiene el algoritmo GRASP. De igual forma se puede obtener una implementación de Tabú Search o Ant Colony Optimization, entre otros. Este fue uno de los objetivos secundarios planeados, ofrecer flexibilidad a posibles extensiones.

El framework fue utilizado para resolver efectivamente el problema de FunsaCoop tomando como referencia la formulación matemática desarrollada. Para la fase constructiva se desarrollaron tres heurísticas constructivas nombradas como Heurística Constructiva por Tiempo de Vulcanizado (HCTV), Heurística Constructiva Alternativa (HCA) y Heurística Constructiva por Demanda (HCD). Cada una de ellas analizan características específicas de una instancia del problema de FunsaCoop permitiendo generar distintos tipos de soluciones. La ventaja radica en la capacidad de exploración del espacio de soluciones por medio de rutinas que analizan soluciones con una estructura específica. Debido a que algunas de las soluciones daban lugar a posibles correcciones, se elaboró un proceso de corrección y ajustes de soluciones para la fase de mejora.

El algoritmo resultante es denominado como "Estimador" y su ejecución genera un planificación de la producción para el problema de FunsaCoop. Los resultados numéricos mostraron que de las 110 instancias del problema definidas, 89 corresponde a soluciones óptimas halladas en menos de 2 segundos. Además se aplicaron varios escenarios de pruebas para validar las soluciones generadas por las heurísticas HCTV, HCA y HCD. No se dio importancia a la optimalidad de las soluciones generadas ya que aquí se aplican técnicas de aproximación. Como forma de validación se comparó las soluciones generadas por Estimador con la planificación de vulcanizado elaborada por los trabajadores de FunsaCoop. Un punto importante es que el Estimador no requiere estimar el parámetro THB del modelo MIP ya que directamente lo calcula al generar soluciones.

En base a los resultados numéricos se vio que el Estimador genera soluciones aproximadas a las óptimas y ya que construye una planificación de la producción se sabe cuanto tiempo se requieren para producir toda la demanda. Esto dio la posibilidad de usar el Estimador como un medio para estimar el parámetro THB de la formulación matemática. Aquí surgió el tercer procedimiento que consistió en combinar técnicas exactas y heurísticas. El objetivo fue mejorar los tiempos empleados por el software CPLEX utilizando un mejor valor del parámetro THB dado por el Estimador. Los resultado mostraron una reducción importante en el tiempo de resolución en varios casos de prueba y la obtención de soluciones óptimas en escenarios que no se obtuvieron.

Como resultado se vio que las tres maneras de resolver el problema de FunsaCoop reflejan características propias de las técnicas empleadas. Para casos chicos del problema las técnicas exactas son las más adecuadas ya que encuentran soluciones óptimas en un tiempo promedio de una hora. Sin embargo para casos grandes emplean mucho tiempo de cómputo debido al análisis exhaustivo que hacen. Por otro lado las técnicas heurísticas, como las aplicadas en el Estimador otorgan soluciones próximas en un tiempo razonable en casos más grande. En

[3]Un framework, entorno de trabajo o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

los casos de pruebas no se superaron los 2 segundos y en los de estrés los 20 segundos. Respecto al Estimador, su capacidad de generar distintas soluciones puede que en términos prácticos no sean las más adecuadas. Sin embargo fueron considerado varios puntos que se aplican a la hora de producir neumáticos haciéndolo así una herramienta capaz de guiar la elaboración de planificaciones.

Adicionalmente se desarrolló una interfaz web para poder ejecutar y visualizar los resultados del Estimador. Aquí se ofrece un medio más amigable y ágil para definir casos del problema de FunsaCoop, ejecutarlo y visualizar la planificación en un diagrama de Gantt. Además es posible guardar y abrir casos desde archivos con un formato definido. Por otro lado el framework elaborado puede ser un medio para resolver otros tipos de problemas de índole similar a la FunsaCoop o distinta. Su capacidad de extensión, el uso de estructuras eficientes y el aprovechamiento del hardware sobre el que se use brindan potencial de resolución.

El resto del documento se organiza de la siguiente manera. El Capítulo 2 formaliza el problema de producción de neumáticos, describiendo datos relevados y planteando su formulación matemática. En el Capítulo 3 se valida la formulación matemática definida para el problema. El Capítulo 4 abarca aspectos de arquitectura, implementación y descripción de la solución informática del problema. En el Capítulo 5 se realiza la experimentación numérica, aplicando escenarios reales obtenidos de FunsaCoop y modificaciones de los mismos, se analiza el comportamiento de las heurísticas propuestas, tiempos de ejecución y calidad de soluciones. En el capítulo 6 se presentan las conclusiones y trabajos futuros. En el Anexo A se tratan los Problemas de Ordenamientos. En el Anexo B se trata Algoritmos, análisis de complejidad y técnicas de resolución. Por último en el Anexo C se muestran las tablas correspondiente a las pruebas realizadas.

CAPÍTULO 2

Definición y formulación del problema

2.1. Introducción

El objetivo de este capítulo es formalizar el problema de planificación en el proceso de vulcanizado. Para lograr entender mejor la problemática en la Sección 2.2 se comienza dando una descripción del proceso de fabricación de neumáticos, abarcando así las etapas requeridas para producirlos, la maquinaria y piezas que intervienen, los puntos claves para mantener los estándares de calidad y los costos que conlleva.

Seguidamente en la Sección 2.3 se describen los datos relevados en las visitas a la fábrica de FunsaCoop. Aquí se mencionan los aspectos principales que influyen en la planificación y producción de neumáticos en el contexto de FunsaCoop. Entre ello se mencionarán las máquinas, tipos de moldes y piezas que se utilizan, como se dispara la producción de FunsaCoop en base a la demanda de los clientes, entre otros. También se describen los problemas que enfrenta FunsaCoop respecto a consumo, gastos y en la forma que planifican la producción.

Posteriormente se realiza un análisis de los datos relevados para determinar los puntos claves a utilizar en la formulación matemática. Se verán las variables, parámetros y restricciones pertinentes al problema. Considerando que la realidad presentada es compleja por todos los factores que intervienen, se describen datos del relevamiento que serán omitidos en la formulación matemática.

En la Sección 2.4 se presenta la formulación matemática. Se verá similitudes y discrepancias con los modelos Discrete Lot-Sizing (DLS) y Job Shop Scheduling (JSP) tratados en el Anexo B. Estos modelos tratan problemas de ordenamiento de tareas para optimizar uso de recursos y otros parámetros, el cual ayudaron para generar una base teórica y práctica. Además se describen trabajos similares aplicados en el proceso de vulcanización utilizados como referencia cuando se formuló el modelo.

2.2. Proceso de fabricación de neumáticos

La producción de neumáticos está compuesta por 3 etapas: 1) mezclado y triturado, 2) ensamblado del neumático y por último 3) curado e inspección (ver figura 2.1). El proceso 1) comienza con la mezcla de ingredientes como caucho, pigmentos, antioxidantes entre otros, el cual son unidos en una mezcladora gigante llamada Banbury, que opera bajo altas temperaturas y presión. Como resultado se tiene todos los ingredientes en un compuesto caliente, negro y pegajoso. El caucho enfriado se corta en tiras que conformará la estructura básica del propio neumático. En la fase de triturado, se preparan otros elementos del neumático donde algunos se recubren con otros tipos de caucho.

Seguidamente 2) el neumático se construye desde dentro hacia fuera. Los elementos textiles, las lonas con cables de acero, los talones, las lonas, las bandas de rodadura y otros componentes se integran en una máquina de construcción de neumáticos. Como resultado se obtiene una “rueda cruda” o “green tire”. A pesar de tener la forma de rueda, no tiene una consistencia suficientemente rígida para llegar a ser una rueda como tal. En la

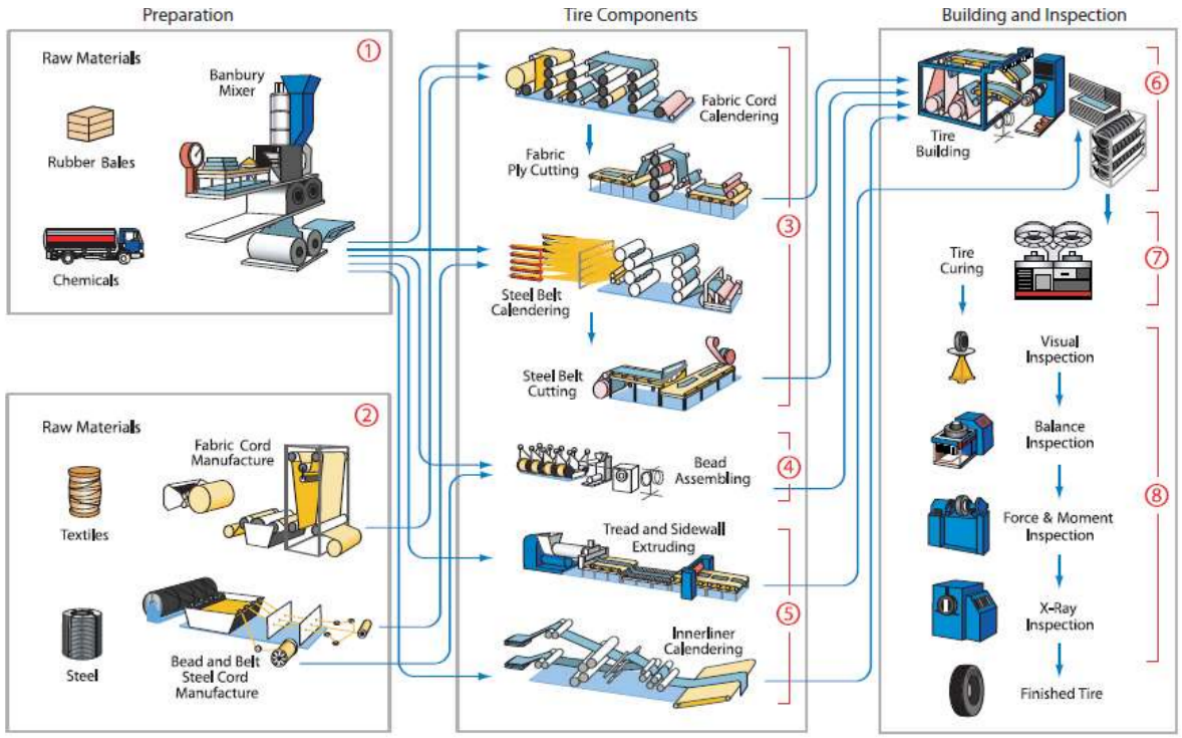


Figura 2.1: Linea de producción de neumáticos (Fuente: Rockwell Automation)

Figura 2.2 se pueden visualizar las ruedas crudas apiladas.

La última 3) etapa tiene como principal objetivo aplicar un proceso que determina la forma final y la rigidez del neumático. Este proceso se denomina "proceso de curado", conocido también por vulcanizado. Para lograrlo se hace uso de dos componentes importantes: 1) los moldes que albergan una green tire para formar el neumático (Figura 2.3), 2) las prensas conocidas también por heaters, que por medio de la presión aplicada comprimen los componentes (Figuras [2.4, 2.5, 2.6]). Cada heater varia en los tipos de moldes capaz de contener, la capacidad de producir más de uno a la vez, el tiempo de cocción, consumo de energía, etc [4]. Luego en la fase de inspección una serie de inspectores formados emplean maquinaria especial para comprobar minuciosamente todos los neumáticos y detectar hasta la más leve imperfección antes de comercializarse. Además del proceso anterior, una muestra de neumáticos se extrae de la línea de producción para someterse a pruebas de rayos X en busca de posibles defectos o debilidades internas.

La vulcanización se considera un proceso critico en la fabricación de neumáticos, la eficacia de trabajo depende de las especificaciones dadas por los clientes y estándares de calidad. Otros factores como el número de heaters en la linea de producción, consumo de combustible y corriente eléctrica tienen influencia en este proceso. De hecho, esta fase consume entre el 60 % y 90 % de la energía disponible en una fabrica [5]. Esta es una de las razones por la que varias plantas destinan parte de su inversión capital en esta fase.

Basado en [5] hay tres elementos en el proceso de vulcanizado claves: tiempo, temperatura y presión. La temperatura y la presión determinan directamente la calidad del neumático. Cuando la temperatura es baja no genera resistencia ni elasticidad suficiente. Por el contrario, altas temperaturas reducen la durabilidad del neumático y la resistencia a las fuerzas que se someten. Si la presión es baja, el neumático tendrá una consistencia suave. En cambio si es alta la vida útil de la maquinaria se reduce y el consumo de energía incrementa.

La fabricación de neumáticos es compleja debido a los factores que se deben considerar y claramente puede generarse un cuello de botella en la etapa de vulcanizado cuando la demanda impone presión y/o no se establece un uso coordinado de la maquinaria. Es necesaria entonces la búsqueda de herramientas que permitan minimizar el tiempo de vulcanizado y maximizar la salida de neumáticos con la mayor calidad posible.

Figura 2.2: Neumáticos crudos (Fuente: UNASEV)

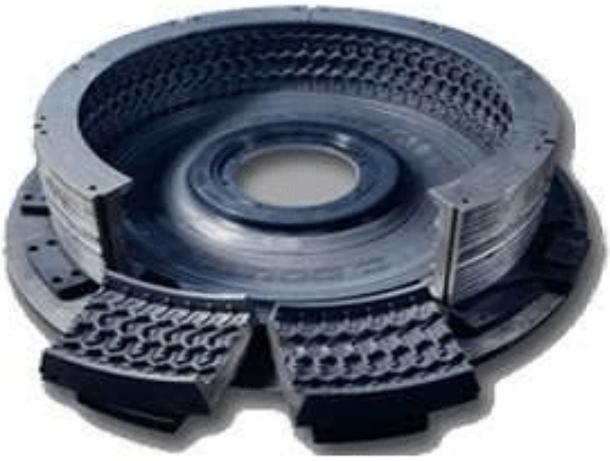


Figura 2.3: Molde de neumático (Fuente: Artículo [6])

Figura 2.4: Prensa de neumático con compartimientos vacíos (Fuente: Pelmar Engineering Ltd.)

Figura 2.5: Prensa de neumático cerrada (Fuente: Pelmar Engineering Ltd.)

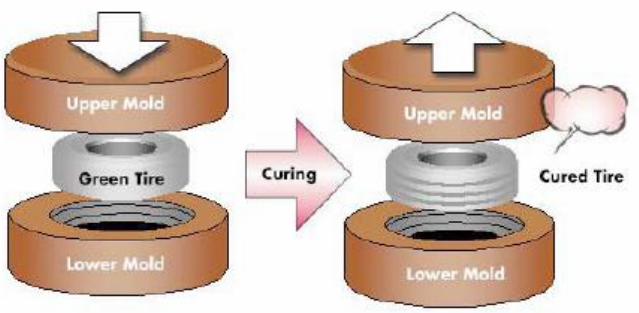


Figura 2.6: Prensa de neumático cerrada (Fuente: sitio web [7])

2.3. Planteo del problema

2.3.1. Entorno de FunsaCoop

La fabrica de neumáticos FunsaCoop abastece al proceso de fabricación con una caldera a leña, el cual su consumo diario es de 25 a 30 toneladas aproximadamente. En la etapa de vulcanización es necesario que la caldera se encuentre a una presión determinada todo el tiempo, mientra que para otras fases la presión puede ser un tercio menor. Dado que los heaters son abastecidos por la caldera, el tiempo ocioso que permanezcan implica un consumo innecesario de leña, desgaste de los heaters, menos producción y más tiempo en realizar la entrega.

FunsaCoop produce nueve tipos de neumáticos, cinco de los cuales son moldes del tipo 42, dos del tipo 55 y dos del tipo 63. Los tipos 42, 55 y 63 hacen referencia al diámetro del neumático. Cada uno de los moldes tiene asociado un tiempo de colocado, un tiempo de vulcanizado y un tiempo en remover del heater. La tarea de colocar un molde incluye configuración y limpieza, tanto en el molde como en el heater utilizado. Se cuenta además con doce heaters, cada uno puede albergar a lo sumo dos moldes del mismo tipo de neumático. Siete de ellos pueden producir solo los neumáticos del tipo 42, tres del 55 y dos del 63. Adicionalmente hay piezas especiales que son compartidas entre moldes. Esto significa que si un molde utiliza estas piezas otro molde que también las necesite no puede ser colocado en un heater hasta que no queden libres. Cada heater puede pasar por varias asignaciones de moldes durante el proceso de fabricación. Si una asignación difiere en algún tipo de molde respecto a la siguiente asignación se deben realizar tareas de mantenimiento y configuración.

Por otro lado la fábrica cuenta con varios clientes en América Latina, uno de los principales es Venezuela dado que genera la mayor demanda. Los pedidos se realizan en lotes de 10,000 neumáticos aproximadamente, teniendo que ser producidos en un periodo de 60 a 90 días. Contando con la cantidad de neumáticos que el cliente solicita, se genera un inventario de green tires y posteriormente se planifica la asignación de los moldes en cada heater. Parte de la planificación consiste en producir asignaciones de moldes el más tiempo posible para reducir los tiempos de configuración y mantenimiento. En la práctica FunsaCoop produce una asignación sin interrupción durante un día, equivalente a tres jornadas laborales. Es importante destacar que actualmente el personal de FunsaCoop realiza esta tarea de forma manual en base a la experiencia generada. Asimismo la planificación efectuada no es estática, hay posibilidades de ser modificada debido a nuevas prioridades de entrega a clientes.

Partiendo con el stock suficiente de green tires, la linea de producción se vé afectada en la etapa de vulcanizado por ser la siguiente fase, generándose así un cuello de botella en este punto. El hecho de no aplicar una buena asignación de producción las fechas de entrega se pueden prolongar y generar perdidas económicas al tener ocioso y en máxima presión la caldera de la fabrica. Por eso motiva afrontar el problema como un Problema de Ordenamiento que minimice la cantidad de periodos requeridos para satisfacer la demanda y maximice el uso de recursos, en especial la caldera cuando está a capacidad máxima. Por otro lado y para no ser de menos el aporte de este proyecto a una cooperativa uruguaya con alcance internacional es otro factor importante y motivador.

2.3.2. Análisis del relevamiento

Para comenzar se plantea como función objetivo minimizar la cantidad de periodos requeridos para satisfacer una demanda dada, esto conlleva la reducción del tiempo ocioso de la caldera y maximizar la vida útil de los moldes y heaters. También es posible reducir la emisión de dióxido de carbono encargada de la contaminación global.

FunsaCoop aplica la política “all-or-nothing” como modo de producción en periodos de un día. Esto significa que durante un día los heaters deben producir solo los tipos de neumáticos que tienen asignados, no se pueden cambiar. Debido a esto se decide discretizar el tiempo en periodos de duración fija. En el marco de FunsaCoop cada día se compone de tres turnos laborales, equivalente a 1440 minutos.

Por otro lado la complejidad de la realidad tratada puede ser muy grande por lo que se decidió acotar el alcance del problema. En el relevamiento se obtuvieron datos que aquí se omiten para simplificar y no abarcar aspectos adicionales y/o externos al problema. Uno de ellos se relaciona con los trabajadores de la planta. Se asume aquí que en todo momento hay trabajadores suficientes para efectuar los cambios de moldes y realizar mantenimientos en las piezas.

También hay otros factores como el tamaño de la planta. Como muchas piezas se pueden encontrar en

distintos lugares se utilizan medios de transporte internos para trasladar las piezas a los heaters. Este traslado no es instantáneo. Aquí se puede asumir que el traslado es instantáneo o puede ser considerado en el tiempo de configuración de la maquinaria. Otro aspecto es la distribución de los heaters en la planta. No todos se encuentran juntos y un problema que se deriva es que varios de ellos se ubican en distintos lugares pero pueden producir mismo tipos de molde. Esto puede generar pérdidas si se deben transportar de forma seguida las piezas en toda la planta. En estos casos los encargados de FunsaCoop generan una planificación tal que los moldes se localicen en una zona específica para evitar traslados frecuentes. Este es otro punto que no se considera como parte del problema. En FunsaCoop se han dado varios casos donde la planificación debe ser realizada nuevamente ya que las prioridades cambian. Esto implica reajustar fechas, cantidades a producir y re configurar máquinas para ajustarse al nuevo plan. Se asume aquí que la planificación es estática, que no se producirá ningún evento que implique reajustar la planificación de la producción.

Como observación a lo mencionado en la Sección 2.3.1 y al relevamiento obtenido de los trabajadores de FunsaCoop se debe considerar los siguientes puntos:

- La cantidad de moldes, heaters y piezas disponibles en cada periodo.
- Compatibilidad entre moldes: solo moldes del mismo diámetro se pueden procesar juntos.
- Compatibilidad entre molde y heater: Los heaters no pueden albergar cualquier tipo de molde, depende del diámetro.
- Para cada tipo de molde (42, 55 y 63) existe distintos tipos de moldes que varían en la ranura del neumático.
- Cuando más frecuentes son los cambios de moldes, más tiempo se utiliza para realizar tareas de mantenimiento y configuración. La política "all or nothing" [1] aplica bien en este escenario ya que a efectos prácticos es mejor producir en lotes grandes.
- Es posible tener heaters sin producir en determinados periodos. Además puede darse que un heater tenga solo un compartimiento ocupado. En la práctica es preferible no hacerlo ya que la presión generada puede generar incidentes y desgastes de maquinaria.
- Ya que es posible procesar en un heater dos moldes del mismo diámetro, la cantidad producida está limitada por el molde con mayor tiempo de vulcanizado.
- Se requiere aplicar configuraciones si el siguiente par moldes a producir en un heater difiere en alguno del par actual.

Restricciones del problema:

1. La cantidad de periodos necesarios debe ser menor a la cantidad de periodos disponibles hasta la fecha de entrega.
2. Los moldes 42, 55 y 63 deben ser procesados por los heaters que pueden albergar los diámetros 42, 55 y 63 respectivamente.
3. Los moldes del mismo diámetro se pueden procesar juntos en un heater compatible.
4. Dado un par de moldes para producir en un heater, este produce a capacidad máxima durante todo un periodo, no hay interrupción.
5. Dado un tipo de molde y un periodo cualquiera, éste puede utilizarse sólo si existen moldes del mismo tipo disponibles y en caso de que requiera piezas deben estar disponibles. Las piezas y moldes utilizados en el periodo actual quedan disponible en el siguiente.
6. Dados dos periodos consecutivos, no se aplica setup si la asignación de moldes se mantiene igual. En caso contrario, a la duración de una jornada laboral se le debe penalizar con cada cambio de molde utilizando los tiempos de quitado y colocación.
7. Para cada molde El stock generado hasta el i-ésimo periodo debe ser menor a su demanda.

Parámetros del problema:

1. Conjuntos de moldes con los que se va a trabajar y la cantidad disponible.

[1] Esta política define que en un periodo de tiempo se produce solo un tipo de artefacto sin interrupción, o no se produce nada.

2. Conjuntos de heaters con los que se va a trabajar.
3. Conjunto de piezas y la cantidad disponible.
4. Conjunto de piezas disponibles y de piezas requeridas por cada molde.
5. La demanda para cada tipo de neumático.
6. El tiempo de vulcanizado, colocado y quitado para cada molde.
7. Duración de la jornada laboral.
8. Máxima cantidad de periodos disponibles.

Variables de problema:

1. Cantidad producida para cada neumático en un periodo dado.
2. Si en un periodo dado existen neumáticos para producir.
3. Si una asignación de moldes se produce o no en un heater y periodo dado.
4. Si dado un heater y un periodo se requiere tiempos de setup.
5. La cantidad de neumáticos producidos en un periodo y heater dada una asignación de moldes.

2.4. Formulación matemática

En FunsaCoop el proceso de fabricación comienza luego de recibida la orden de los clientes. A esta forma de trabajo se le conoce como "make-to-order" y corresponde a un modo de fabricación. Muchas características del ambiente de trabajo se asemejan a las del modelo Discrete Lot-Sizing Problem (ver Anexo A). Por ejemplo la demanda es determinista y variable, el plan de producción se ejecuta sobre un horizonte de tiempo finito dividido en jornadas laborales, se aplica la política "all-or-nothing", se deben manejar costos de configuración y mantenimiento de maquinaria e inventario. No obstante, existen características que extienden a DLSP: 1) es de carácter multi-resource (se cuenta con varios tipos de heaters) y 2) multi-item (los heaters son capaces de producir varios tipos de neumáticos) [8].

Por otra parte existen similitudes con el modelo Job Shop Scheduling Problem (JSP) visto en el Anexo A, el cual corresponde a un tipo de problemas de planificación de tareas. Uno de sus principales objetivos es reducir el tiempo total de realización (makespan). Para el caso de FunsaCoop corresponde a minimizar el tiempo total de producción. Su similitud se debe a que se cuenta con un conjunto de máquinas (heaters) y una tarea de vulcanizado, donde cada tarea se compone por una secuencia ordenada de operaciones que operan durante un periodo de tiempo (asignaciones a producir). Entre las restricciones similares se tiene que una operación no puede ser interrumpida cuando utiliza una máquina (heater). Además una máquina puede ser utilizada solo por una operación. No obstante se diferencia con el caso de FunsaCoop al no fragmentar la producción en periodos fijos de tiempo, como jornadas laborales.

En la literatura se pueden encontrar trabajos relacionados para resolver el problema de vulcanizado. Un caso particular es el estudiado por Degraeve and Schrage [1], el cual resuelven este problema en una fábrica que utiliza una técnica de vulcanizado poco aplicada en la actualidad. Una variante más general y que se asemeja al problema de FunsaCoop es el estudio realizado por Jans y Zeger [2], el cual considera en el modelo distintos tipos de neumáticos, heaters, compatibilidad entre molde-heater, costos de inventario y configuración. El proceso de fabricación se dispara en base a la demanda de los clientes (make-to-order). Otra variante es el trabajo realizado por Shengping y Yang [3] el cual no aplican DLSP, sin embargo las restricciones empleadas se asemejan a la realidad de FunsaCoop. El problema de FunsaCoop se asemeja a los estudios nombrados y además añade complejidad dado que se debe considerar restricciones de compatibilidad entre moldes, la existencia de recursos compartidos, disponibilidad de varios tipos de moldes y heaters, costos y tiempos de configuración.

Una característica que comparten todos los trabajos realizados es la omisión de ciertos aspectos de la realidad. Cada uno de ellos abarcan una visión o problemática específica omitiendo otros factores que influyen en la complejidad del problema, o que se podrían tratar como un problema distinto. En la formulación matemática desarrollada en este proyecto se pasan por alto características como los tiempos de traslado, la planificación dinámica, cantidad de trabajadores y la distribución de piezas en la fábrica.

A grandes rasgos la formulación elaborada sigue la siguiente forma.

Minimizar makespan

sujeto a:

(i) Restricciones de requerimiento de heater y continuidad de producción en los periodos.

(ii) Restricciones de capacidad de producción en heaters.

(iii) Restricciones de inventariado.

(iv) Restricciones de capacidad y disponibilidad de moldes y piezas.

(v) Restricciones de precedencia y setups.

(vi) Restricciones de dominios de variables.

La función objetivo busca una planificación que minimice el tiempo total de producción (makespan). Las restricciones en (i) determinan cuando un heater y un periodo son requeridos. También se incluyen otras restricciones que exigen continuidad en los periodos utilizados, esto es no dejar periodos intermedios sin utilizar. En (ii) se restringe la capacidad de producción en los heaters dado que solo pueden albergar 2 moldes. En (iii) se definen restricciones que aseguren que la demanda de los clientes se satisfaga. En (iv) se controla la disponibilidad y uso de moldes y piezas compartidas por periodos. En (v) se determina si entre asignaciones consecutivas cuantos setups se necesitan para colocar y remover moldes por periodo. Por último en (vi) se definen los dominios de las variables.

La complejidad que se trata puede ser muy grande debido a la cantidad de combinaciones que pueden haber, incluso para escenarios chicos. Por ejemplo suponiendo un escenario con los parámetros de la Tabla 2.1 se pueden generar varias soluciones que a veces intuitivamente parecen correctas pero no. La Tabla 2.2 muestra 3 soluciones posibles donde la mejor requiere de 3 periodos. Las soluciones "AB" y "ABAAAB" requieren de 4 periodos sin embargo generan un exceso de neumáticos importante, en especial la primera.

Parámetro	Valor
Duración de jornada (ϕ)	$8hrs$
#heaters	1 (h1)
#moldes	2 (A, B)
Tiempo de setup	$s_A = s_B = 0{,}5hr$
#moldes disponibles	$A = 1, B = 1$
#demanda	$dm_A = 15, dm_B = 8$
piezas compartidas	no
tiempo de vulcanizado	$tv_A = 1hr, tv_B = 2hr$

Tabla 2.1: Ejemplo de instancia chica del problema.

Periodo	Solución "AB"	Solución "ABAAAB"	Solución "AAABBB"
1	setup A,B (1hr) 3 ciclos A+B (6hrs)	setup A,B (1hr) 3 ciclos A+B (6hrs)	setup A,A (1hr) 7 ciclos A+A (7hrs)
stock	3A, 3B	3A, 3B	14A, 0B
2	4 ciclos A+B (8hrs)	setup A (0.5hr) 7 ciclos A+B (7hrs)	setup B (0.5hr) 3 ciclos A+B (6hrs)
stock	7A, 7B	10A, 3B	17A, 3B
3	4 ciclos A+B (8hrs)	setup B (0.5hr) 3 ciclos A+B (6hrs)	setup B (0.5hr) 3 ciclos B+B (6hrs)
stock	11A, 11B	13A, 6B	17A, 9B
4	4 ciclos A+B (8hrs)	4 ciclos A+B (8hrs)	
stock	15A, 15B	17A, 10B	

Tabla 2.2: Posibles soluciones. La óptima es la solución "AAABBB"

A continuación se presenta la formulación matemática el cual está basada en la presentada por Degraeve [2], adicionando las restricciones específicas al problema de FunsaCoop.

Conjuntos

$0 =$ Molde "vacío"o ficticio
$M =$ Conjunto de moldes, índices $i, j, l \in \{1...|M|\}$
$MExt = \{0\} \cup M =$ Se extiende M con el molde ficticio
$H =$ Conjunto de máquinas de vulcanizado (heater), índices $k, k' \in \{1...|H|\}$
$P =$ Conjunto de períodos, indice $t \in \{1...|P|\}$
$PExt =$ Conjunto P extendido con el periodo 0, índice $t' \in \{0\} \cup P$
$Q =$ Conjunto de piezas disponibles, índice $q \in 1...|Q|$
$C =$ Conjunto de pares molde-heater compatibles (i, k).
$CExt = C \cup \{(0, k) \mid k \in H\}$, el molde ficticio es compatible con cada heater.
$MC =$ Conjunto de moldes que se pueden procesar juntos (i, j) con $i \leq j$.
$MCExt = MC \cup \{(0, i) \mid i \in M\}$, el molde ficticio es compatible con los demás moldes.

Parámetros

$THB =$ Cantidad de periodos disponibles, con $PN = |P|$
$dm_i =$ Demanda de la rueda i (unidades).
$tc_i =$ Tiempo necesario para colocar el molde i (minutos).
$tq_i =$ Tiempo necesario para quitar el molde i (minutos).
$tv_{i,k} =$ Tiempo de vulcanizado del molde i en el heater k (minutos).
$\Omega =$ Duración de un período (minutos).
$nm_i =$ Cantidad de moldes del tipo i disponibles.
$np_q =$ Cantidad de piezas del tipo q disponibles.
$rq_{iq} =$ Parámetro binario que indica si el molde i require la pieza q.
$xInit_{ik} =$ Cantidad inicial de moldes del tipo i utilizados en el heater k durante el periodo inicial, con $xInit_{ikt} \in \{0, 1, 2\}$.
$prdInit_i =$ Stock inicial del neumático tipo i.

Conjuntos auxiliares

$$T = \{(i,j,k) \mid (i,j) \in MC \wedge \{(i,k),(j,k)\} \subseteq C \wedge i \leq j\}$$
$$= \text{Moldes compatibles } (i,j) \text{ que se pueden procesar en la máquina } k.$$
$$T_i = \{(j,k) \mid (i,j,k) \in T\}$$
$$T_i' = \{(j,k) \mid (j,i,k) \in T\}$$
$$PC_q = \{i \mid rq_{iq} = 1, \forall i \in M\}, \text{ con } q \in Q$$
$$= \text{Moldes que comparten la pieza } q.$$
$$TExt = \{(i,j,k) \mid (i,j) \in MExt \wedge \{(i,k),(j,k)\} \subseteq CExt \wedge i \leq j\}$$
Conjunto análogo a T adicionando el uso del molde ficticio.
$$TExt_i = \{(j,k) \mid (i,j,k) \in TExt\}$$
$$TExt_i' = \{(j,k) \mid (j,i,k) \in TExt\}$$
$$HExt_k = \{(i,j) \mid (i,j,k) \in TExt\}$$

Variables

$x_{ikt'}$ = Cantidad de moldes del tipo i utilizados en el heater k durante el periodo t', con $x_{ikt'} \in \{0,1,2\}$.

y_{ikt} = Cantidad de "start-ups"necesarios para producir el molde i en el heater k durante el periodo t, con $y_{ikt} \in \{0,1,2\}$.

y'_{lkt} = Cantidad de moldes l utilizados en el periodo anterior y que es necesario quitar para utilizar el heater k en el periodo t, con $y'_{lkt} \in \{0,1,2\}$.

z_{ijkt} = Variable binaria que indica si los moldes (i,j) son producidos en el heater k durante el periodo t.

w_t = Variable binaria que indica si en el periodo t existe algún heater en uso.

μ_{ijkt} = Cantidad de unidades producidas del neumático de tipo i cuando se procesa junto a j en el heater k durante el periodo t.

prd_{it} = Cantidad producida del neumático tipo i en el periodo t.

Planteo

minimizar
$$\sum_{t \in P} w_t \tag{2.1}$$

s.a
$$w_t \leq w_{t-1}, \forall t \in P \setminus \{1\} \tag{2.2}$$
$$w_t \geq \frac{1}{2|H|} \sum_{(i,j,k)|i \in TExt} z_{ijkt}, \forall t \in P \tag{2.3}$$
$$\sum_{(i,j) \in HExt_k} z_{ijkt} \leq 1, \forall k \in H, \forall t \in P \tag{2.4}$$
$$\mu_{0jkt} \leq \frac{(\Omega - y_{jkt}.tc_j - \sum_{l \in M} y'_{lkt}.tq_l)}{tv_{jk}}, \forall (0,j,k) \in T, \forall t \in P \tag{2.5}$$
$$\mu_{ijkt} \leq \frac{(\Omega - y_{ikt}.tc_i - y_{jkt}.tc_j - \sum_{l \in M} y'_{lkt}.tq_l)}{max(tv_{ik}, tv_{jk})}, \forall (i,j,k) \in T, i < j, \forall t \in P \tag{2.6}$$
$$\mu_{iikt} \leq \frac{(\Omega - y_{ikt}.tc_i - \sum_{l \in M} y'_{lkt}.tq_l)}{tv_{ik}}, \forall (i,i,k) \in T, \forall t \in P \tag{2.7}$$
$$\mu_{ijkt} \leq \Omega.z_{ijkt}, \forall (i,j,k) \in TExt, \forall t \in P \tag{2.8}$$
$$prd_{it} = \sum_{(j,k) \in T_i} \mu_{ijkt} + \sum_{(j,k) \in TExt'_i} \mu_{jikt}, \forall i \in M, \forall t \in P \tag{2.9}$$
$$prdInit_i + \sum_{t \in P} prd_{i,t} \geq dm_i, \forall i \in M \tag{2.10}$$
$$x_{ikt} = \sum_{(j,k) \in T_i} z_{ijkt} + \sum_{(j,k) \in TExt'_i} z_{jikt}, \forall i \in M, \forall k \in H, \forall t \in P \tag{2.11}$$
$$\sum_{(i,k) \in C} x_{ikt} \leq nm_i, \forall i \in M, \forall t \in P \tag{2.12}$$
$$\sum_{i \in PC_q} x_{ikt} \leq np_q, \forall q \in Q, \forall k \in H, \forall t \in P \tag{2.13}$$
$$x_{ik0} = xInit_{i,k}, \forall i \in MExt, \forall k \in H \tag{2.14}$$
$$y_{ikt} \geq x_{ikt} - x_{ik(t-1)}, \forall i \in M, \forall k \in H, \forall t \in P \tag{2.15}$$
$$y'_{ikt} \geq x_{ik(t-1)} - x_{ikt}, \forall i \in M, \forall k \in H, \forall t \in P \tag{2.16}$$
$$x_{ikt'}, y_{ikt}, y'_{ikt} \in \{0,1,2\}, i \in M, k \in H, t \in P, t' \in PExt \tag{2.17}$$
$$\mu_{ikt} \in N, i \in MExt, k \in H, t \in PExt \tag{2.18}$$
$$w_t \in \{0,1\}, t \in P \tag{2.19}$$
$$prd_{it} \in N, i \in M, t \in P \tag{2.20}$$
$$z_{ijkt} \in \{0,1\}, i \in MExt, j \in M, k \in H, t \in PExt \tag{2.21}$$

La función objetivo 2.1 buscar minimizar el makespan, esto es el tiempo total de producción. Implícitamente se están aprovechando más los recursos, dado que la caldera mantiene menos tiempo ocioso y el uso de maquinaria y piezas se maximiza.

La restricción 2.2 impone continuidad en el uso de periodos, evitando así dejar periodos huecos.

La restricción 2.3 especifica que un periodo es requerido si existe un heater en uso y hay moldes para producir en dicho periodo.

La restricción 2.4 establece que en cada periodo un heater es capaz de producir a lo sumo un par de moldes.

Las restricciones 2.5, 2.6, 2.7 y 2.8 determinan la cantidad de neumáticos que se pueden producir en un heater y en un periodo dado en función de los moldes asignados y los utilizados en el periodo anterior. Aquí se incluye la penalización en la duración de la jornada laboral cuando los moldes del periodo anterior difieren respecto al actual, dado que son requeridas tareas de mantenimiento y configuración.

Las restricciones 2.9 y 2.10 determinan el inventario y aseguran que la demanda de cada tipo de neumático se satisfaga.

La restricción 2.11 establece la cantidad del molde tipo i que se deben colocar en el heater k durante el

periodo t. Esta variable se determina en función de z, que indica si existe una asignación de producción para el molde i.

Las restricciones 2.12 y 2.13 limitan el uso de moldes en cada periodo en función de la cantidad de moldes y piezas que cuenta la fábrica.

La restricción 2.14 inicializa los heaters con una determinada cantidad de moldes. Esto representa el estado inicial de los heaters cuando comienza la producción.

La restricción 2.15 determina para un molde la cantidad de start-ups necesarios en un heater en función de la cantidad del mismo molde utilizado en el periodo anterior y los requeridos en el actual. Por ejemplo si en el periodo anterior se utiliza un molde del tipo α y en el actual son utilizados dos moldes tipo α, entonces en el periodo actual se requiere un start-up para aplicar en el heater. De manera análoga la restricción 2.16 determina cuantos moldes difieren respecto al periodo anterior para calcular la cantidad de moldes que hay que remover del heater. Por último las restricciones 2.17, 2.18, 2.20 y 2.21 definen el dominio de cada variable.

Como se describió en la Sección 2.3.2 la formulación se simplificó para no abarcar otros aspectos de la realidad, el cual conllevaría una complejidad aún mayor. Aquí se genera un ordenamiento de la producción considerando capacidad y disponibilidad de heaters, moldes y piezas y penalización del tiempo de producción al aplicar configuraciones.

Para simplificar aspectos del problema enfrentado se asume que hay suficientes trabajadores disponibles para realizar tareas de mantenimiento y configuración. La ubicación de los heaters, moldes y piezas en la fábrica implica tiempo gastado en traslado, el cual se asume instantáneo. Cabe destacar que los trabajadores de FunsaCoop crean una planificación para que ciertas piezas y moldes se mantengan cercanos, justamente para reducir el tiempo de traslado. Otro aspecto es que la planificación es estática, se asume que no habrá eventos externos que modifiquen la planificación. Otro punto es la capacidad de los heaters, se asume que todos son capaces de albergar a lo sumo dos moldes.

Algunos de los puntos asumidos se pueden considerar como parte de los parámetros del problema. Por ejemplo los tiempos de traslado de piezas y moldes se pueden incluir en los tiempo de colocación. En caso que la planta cuente con un heater que solo alberga un molde se puede modelar agregando un molde que solo sea compatible con ese heater.

En cuanto al tamaño del modelo matemático se estima que cuenta con $O(|M|^2 \times |H| \times |THB|)$ variables y $O((|M|^2 \times |H| + |Q|) \times |THB|)$ restricciones. Se puede ver que el parámetro THB determina la cantidad de periodos disponibles para realizar una entrega influye en el tamaño del modelo. Este valor se puede ver como una cota superior de la función objetivo, lo cual valores grandes implica el uso de más variables y restricciones. Valores cercanos a la solución exacta conlleva una cantidad más razonable. Se verá en los siguientes capítulos como este parámetro se estima para mejorar el tiempo de resolución del problema.

CAPÍTULO 3

Validación del modelo

3.1. Introducción

El objetivo de este apartado es validar la formulación matemática presentada en el Capítulo 2. A través de un plan de pruebas se proponen casos básicos y espaciales para revisar y examinar la estructura de las soluciones dadas por la formulación. La combinación de los casos sencillos fueron la forma de corroborar casos más complejos.

La formulación se codificó en AMPL, un lenguaje destinado para representar modelos matemáticos. También se utilizó el software CPLEX que gracias a este se puede resolver el modelo codificado en AMPL y obtener soluciones óptimas para instancias del problema.nEn total se elaboraron 20 casos de pruebas compuestos por 14 básicos y 6 espaciales. Cada uno se evaluó de forma manual para saber que solución sería la correcta y se corroboró con la solución dada por CPLEX. La elaboración de casos de prueba ayudaron para validar y encontrar fallas en la formulación matemática como en la codificación AMPL. Entre los casos definidos se prueba instancias que validan compatibilidad entre moldes, moldes y heater, disponibilidad de moldes y piezas, entre otros. Se dieron muchos casos donde la solución obtenida manualmente no coincide con la dada por CPLEX pero se llegó al mismo valor óptimo.

Este capítulo se organiza de la siguiente manera. En la Sección 3.2 se describen aspectos de la codificación del modelo y la forma de resolución a través de CPLEX. En la Sección 3.3 se describen los casos de pruebas aplicados. Para cada uno se realiza una descripción de la instancia, el resultado espero y comentarios respecto a soluciones alternativas. Por último a modo de resumen se comentan aspectos generales, como errores encontrados y estructuras de las soluciones, entre otros.

3.2. Codificación y resolución de formulación matemática

Para poder realizar la validación el modelo se decidió codificarlo al lenguaje AMPL (A Mathematical Programming Language) el cual está dirigido a la construcción y resolución de modelos de optimización, fundamentalmente de Programación Lineal, Programación Entera y Programación No Lineal.

Para resolverlo fue necesario buscar softwares que interpreten el lenguaje y lo resuelvan. Aquí se cuenta con una gran variedad de solvers comerciales y de código abierto que pueden utilizarse. De los que destacan es el solver comercial CPLEX desarrollado por IBM. GLPK es otro solver pero de código abierto. Se decidió utilizar CPLEX en vez de GLPK debido a la complejidad y tamaño del modelo presentado. En muchas de las instancias probadas GLPK no fue capaz de encontrar una solución en un tiempo razonable, o directamente no podía resolver el problema. Mientras que CPLEX tuvo una mejor respuesta al hallar soluciones óptimas en menos tiempo. Estos dos solvers buscan en un modelo matemático un solución óptima y aplican una variedad de técnicas exactas para hallarla. Haber utilizado estas herramientas permitió corroborar los resultados al resolver las instancias manualmente.

Recursos de cómputo	
CPU	Intel quad-core 3.0Ghz
Memoria	16Gb
Sistema Operativo	Windows 10 (x64)
CPLEX	12.6.0.0

Tabla 3.2: Hardware y software utilizados

3.3. Casos de pruebas

En la formulación matemática elaborada en la Sección 2.4 se pueden extraer ciertos parámetros que influyen en las decisiones tomadas para generar un ordenamiento de la producción. Estos parámetros son la cantidad de heaters, tipos de molde, piezas compartidas, demanda de cada molde y los tiempos de vulcanizado.

La cantidad de heaters influye dado que al contar con más lineas de producción el modelo debe determinar las asignaciones que más provecho saquen de la maquinaria disponible. La cantidad de cada tipo de molde y piezas compartidas influye a la hora de decidir cuantos moldes se pueden producir en paralelo. Los parámetros de demanda, tiempo de vulcanizado y cantidad de cada tipo de molde afecta a la hora de decidir como agrupar moldes en pares. Estos parámetros son utilizados para definir cada caso de prueba:

- Cantidad de heaters.
- Cantidad de moldes.
- Disponibilidad por cada tipo de molde.
- Si la demanda es igual o no.
- Si existen piezas compartidas entre moldes.

Por último los parámetros de tiempo de colocado y quitado son utilizados como constante en los casos de prueba. Para generar diversidad de las pruebas se decide probar con uno y dos tipos de moldes, variando si las demandas son iguales o no, y en algunos casos considerar si se comparten piezas entre los moldes. Se asume que en cada caso todos los moldes son compatibles entre sí. A continuación se presentan los casos de pruebas describiendo los casos propiamente, objetivos y salida esperada de la prueba. Se comienza con los casos sencillos definidos en el esquema de la Figura 3.1, posteriormente se presentan los casos especiales.

En las Tablas 3.2 y 3.1 se visualizan los valores utilizados como parámetro globales y los recursos de cómputos para correr las pruebas respectivamente.

Parámetro	Valor
Duración de jornada (ϕ)	60u
Tiempo colocado 1 (tc1)	5u
Tiempo colocado 2 (tc2)	5u
Tiempo quitado 1 (tq1)	5u
Tiempo quitado 2 (tq2)	5u

Tabla 3.1: Parámetros globales a todos los casos de validación

3.3.1. Casos simples

Casos 1 y 2

Descripción: Estos dos casos son los más sencillos que se pueden aplicar al modelo, los valores utilizados para cada uno se muestran en la Tabla 3.3 y se espera en ambos la misma salida. Se espera que cada asignación aplicada al único heater utilice todos los moldes disponibles. Por ende se requiere en el Caso 1 y 2 utilizar 4 y 2 periodos respectivamente. En ambos escenarios el modelo responde con la salida esperada.

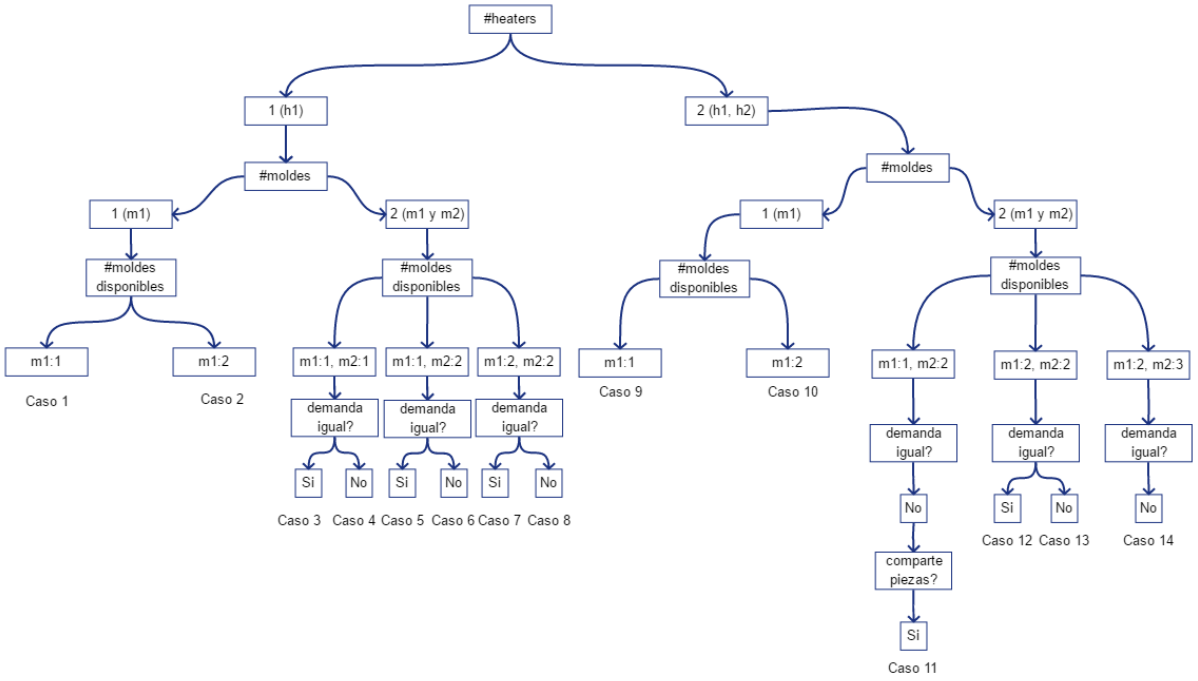


Figura 3.1: Plan de pruebas (no incluye casos de especiales)

Parámetros:

Parámetro	Valor
#heaters	1
#moldes	1 (m1)
#moldes disponibles	m1=1
#demanda	dm1=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u

Parámetro	Valor
#heaters	1
#moldes	1 (m1)
#moldes disponibles	m1=2
#demanda	dm1=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u

Tabla 3.3: Parámetros de los Casos 1 y 2.

Casos 3 y 4

Descripción: Estos casos cuentan con dos tipos de moldes ($m1$, $m2$) compatibles entre sí y un heater ($h1$); para cada tipo de molde se dispone de una sola unidad. El objetivo es probar que el modelo es capaz de agrupar moldes compatibles entre sí.

Las instancias dadas en las Tablas 3.4 tiene como mejor asignación producir los dos tipos en el mismo heater durante los periodos necesarios; esta asignación es independiente si las demanda son iguales o no. El modelo responde adecuadamente respecto a la salida esperada en ambos casos.

Parámetros:

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=1, m2=1
#demanda	dm1=20u, dm2=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=1, m2=1
#demanda	m1=20u, m2=37u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.4: Parámetros de los Casos 3 y 4.

Casos 5 y 6

Descripción: Análogo al Caso 1 y 2,se cuenta ahora con un molde más del tipo $m2$. Además del objetivo de los Casos 3 y 4 se quiere probar que el modelo no asigne más moldes de los que un heater pueda soportar. Las instancias de la Tabla 3.5 generan una nueva estrategia a utilizar: el modelo puede seleccionar producir primero una tanda de la asignación $(m2, m2)$ y luego completar la demanda de ambos moldes asignando el par $(m1, m2)$.

En el caso que las demandas son distintas esta asignación es la más adecuada y requiere de 8 periodos; para el caso que las demanda son iguales la mejor asignación es la $(m1, m2)$, requiriendo 6 periodos. Gracias a esta prueba, se encontró un error en los cálculos de producción cuando la asignación utiliza moldes del mismo tipo; se producía una cantidad menor a la correcta.

Parámetros:

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=1, m2=2
#demanda	dm1=20u, dm2=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=1, m2=2
#demanda	dm1=20u, dm2=37u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.5: Parámetros de los Casos 5 y 6.

Casos 7 y 8

Descripción: En estos escenario se cuenta con dos moldes de cada tipo; como estrategia una buena decisión es usar dos moldes del mismo tipo en un heater para lograr producir lo más posible. El objetivo de estos casos es evaluar la capacidad de aprovechar moldes del mismo tipo cuando se encuentra disponible. Independiente de la demanda esta es la mejor estrategia a aplicar: producir la asignación $(m1, m1)$ y luego la $(m2, m2)$, por ende se requieren 5 y 7 periodos para los Casos 7 y 8 respectivamente.

Mediante este caso se encontró un error al calcular el tiempo disponible de un periodo cuando la siguiente asignación de moldes difiere en algún tipo; si disponía más de lo debido, en consecuencia se requerían erróneamente menos periodos para generar todo el inventario.

Parámetros:

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=2, m2=2
#demanda	dm1=20u, dm2=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1, m2)
#moldes disponibles	m1=2, m2=2
#demanda	dm1=20u, dm2=37u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.6: Parámetros de los Casos 7 y 8.

Casos 9 y 10

Descripción: A partir de estos escenarios se cuenta con dos heaters ($h1$ y $h2$). Dadas las instancias en la Tabla 3.7 se cuenta con un tipo de molde; se pretende probar que el modelo se limita a utilizar la cantidad de moldes disponibles por más compartimientos que se cuenten. Existen varias soluciones óptimas en estos escenarios, pero una buena estrategia es producir solamente en un heater (para evitar tiempos de setup), por ende se requiere para la instancia 9 y 10, 4 y 2 periodos respectivamente. El modelo responde de la manera mencionada en ambas instancias.

Parámetros:

Parámetro	Valor
#heaters	2 (h1, h2)
#moldes	1 (m1)
#moldes disponibles	m1=1
#demanda	dm1=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u

Parámetro	Valor
#heaters	2 (h1, h2)
#moldes	1 (m1)
#moldes disponibles	m1=2
#demanda	dm1=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u

Tabla 3.7: Parámetros de los Casos 9 y 10.

Casos 11

Descripción: El objetivo de este caso es validar la intervención de piezas compartidas entre moldes. Las piezas compartidas producen la formación de un cuello de botella ya que para ser utilizadas se debe esperar a que queden disponibles.

Dada la instancia definida en la Tabla 3.8, el uso de piezas compartidas implica que por más heaters y moldes que se tengan, solo se puede producir un molde a la vez. Una salida posible es la que se muestra en la Figura 3.2, requiriendo 14 periodos, una alternativa valida se muestra en la Figura 3.3.

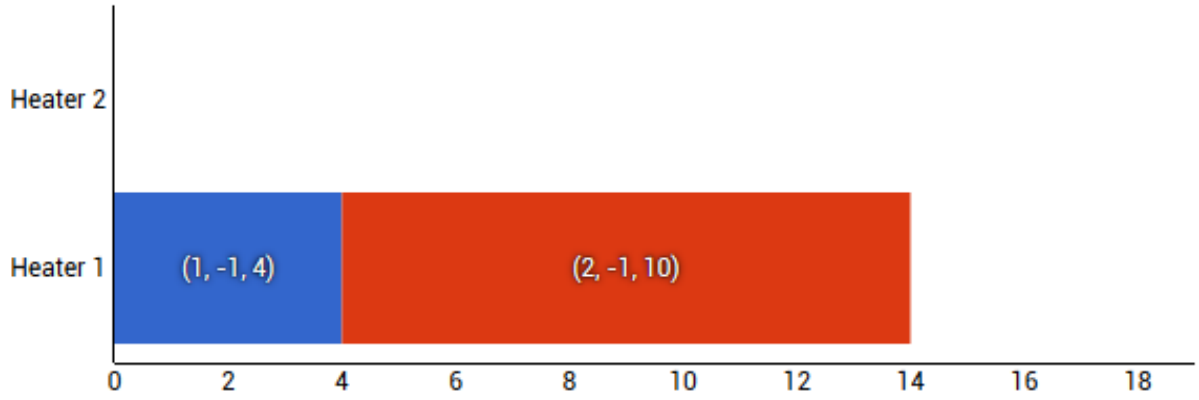


Figura 3.2: Posible solución del Caso 11

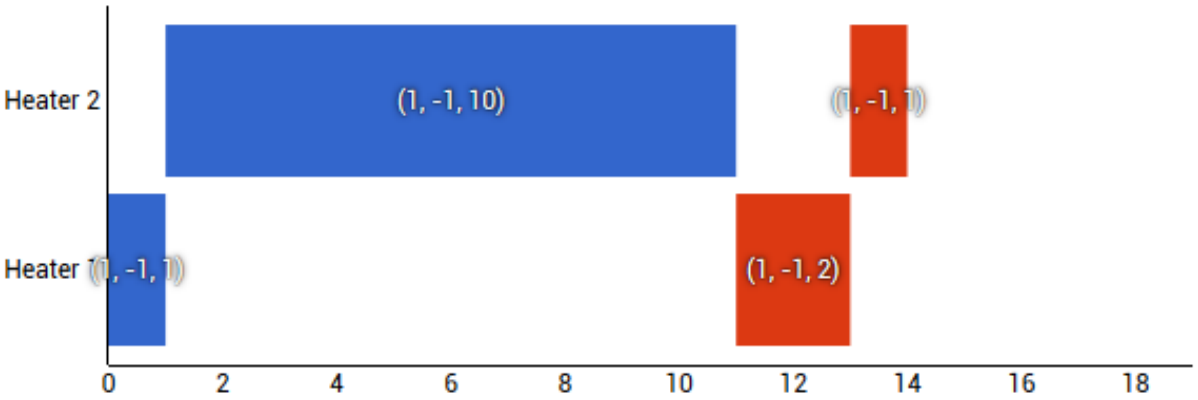


Figura 3.3: Solución del Caso 11 obtenida por el modelo

Parámetros:

Parámetro	Valor
#heaters	2 (h1 y h2)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=1 y m2=2
#demanda	dm1=20u y dm2=37u
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.8: Parámetros del Caso 11.

Casos 12 y 13

Descripción: Estos escenarios contienen los heaters $h1, h2$ y dos tipos de moldes $m1, m2$, contando con dos unidades de capa tipo (ver Tabla 3.9). El objetivo es probar que el modelo aprovecha todos los recursos cuando se encuentran disponibles. Al contar con más de un molde del mismo tipo es preferible producir pares del mismo tipo ya que el tiempo de vulcanizado es el mismo y es posible producir más en menos tiempo. Se espera para ambos casos que la solución dada por el modelo sean las mostradas en las Figuras (3.4 y 3.5).

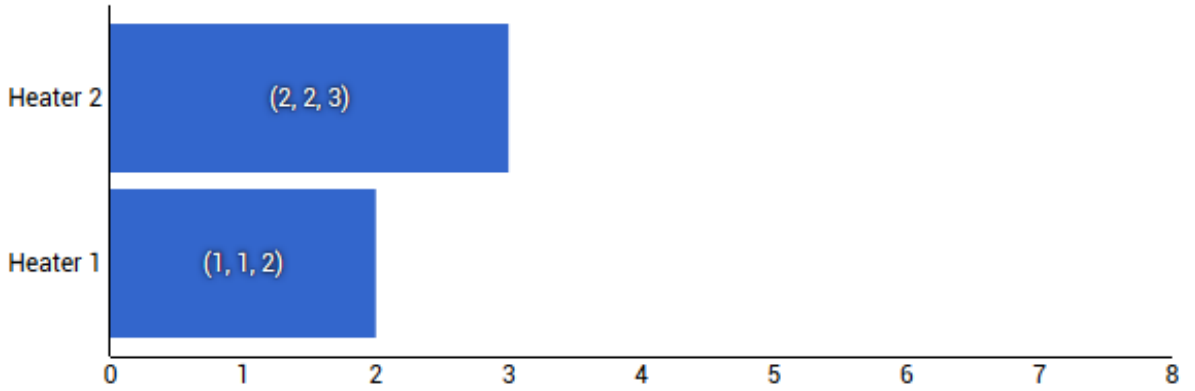


Figura 3.4: Solución del Caso 12

Parámetros:

Parámetro	Valor
#heaters	2 (h1 y h2)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=2 y m2=2
#demanda	dm1=20u y dm2=20u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Parámetro	Valor
#heaters	2 (h1 y h2)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=2 y m2=2
#demanda	dm1=20u y dm2=37u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.9: Parámetros de los Casos 12 y 13.

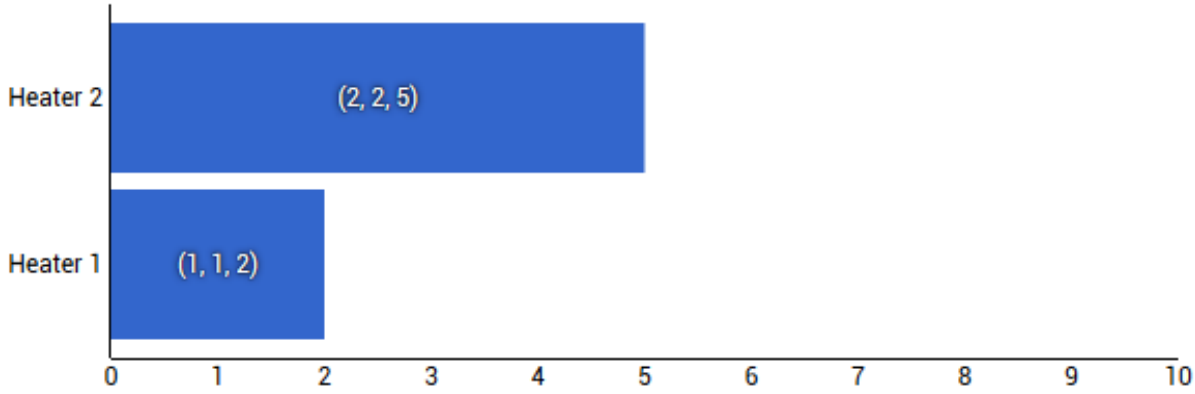


Figura 3.5: Solución del Caso 13

Caso 14

Descripción: En este escenario se cuenta con 5 moldes y 2 heaters (4 compartimientos), se pretende probar que el modelo es capaz de seleccionar una asignación donde se mantenga la mayor cantidad de moldes produciendo en paralelo para así reducir la cantidad de periodos utilizados. Una posible solución es la que se muestra en la Figura 3.6 donde se requiere 4 periodos; una alternativa dada por el modelo se puede ver en la Figura 3.7

Parámetros:

Parámetro	Valor
#heaters	2 (h1 y h2)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=2 y m2=3
#demanda	dm1=20u y dm2=37u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.10: Parámetros del Caso 15

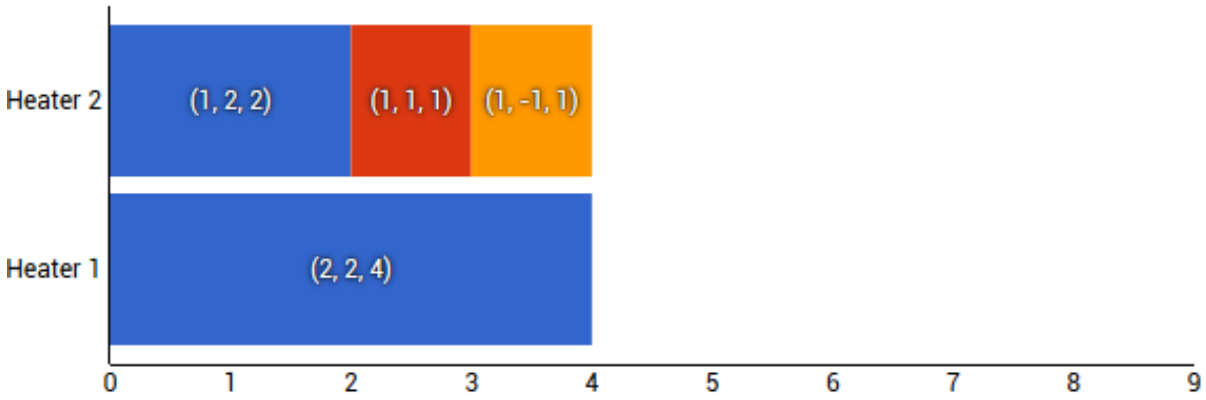


Figura 3.6: Posible solución del Caso 14

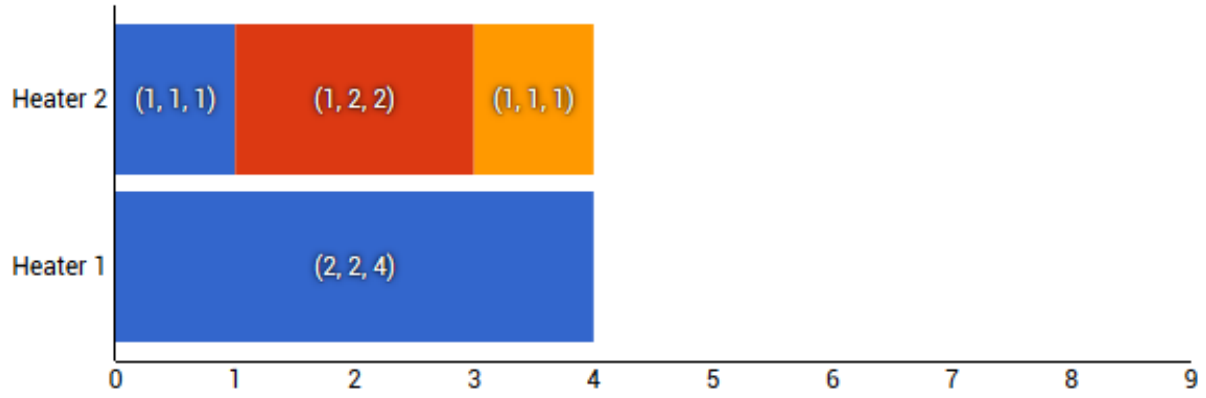


Figura 3.7: Solución del Caso 14 dado por el modelo

3.3.2. Casos especiales

Caso 15

Descripción: Dada la instancia de la Tabla 3.11 el objetivo de prueba es verificar que el modelo no exceda la producción de neumáticos al tener demandas distintas en dimensión. La instancia muestra que la demanda de un molde es de 20 unidades y la del otro 1000 unidades. Una solución errónea es tratar de producir el par hasta satisfacer la demanda de las 1000 unidades, se produciría en exceso de producción del molde que solo requiere de 20 unidades. Inicialmente el modelo generaba este tipo de soluciones, este caso ayudó detectar el problema. Una solución valida es la que se visualiza en la Figura 3.8; la alternativa dada por el modelo corregido se muestra en la Figura 3.9.

Parámetros:

Parámetro	Valor
#heaters	1 (h1)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=1 y m2=1
#demanda	dm1=20u y m2=1000u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.11: Parámetros del Caso 15

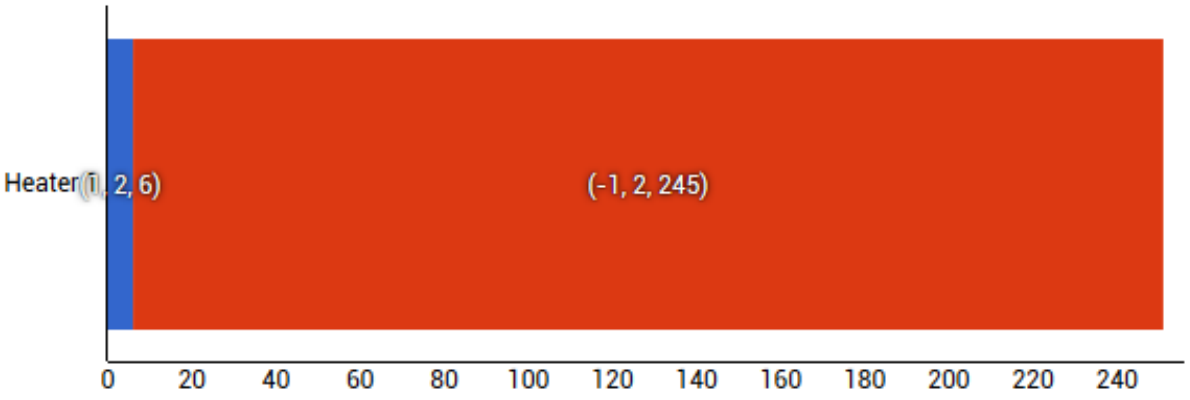


Figura 3.8: Posible solución del Caso 15

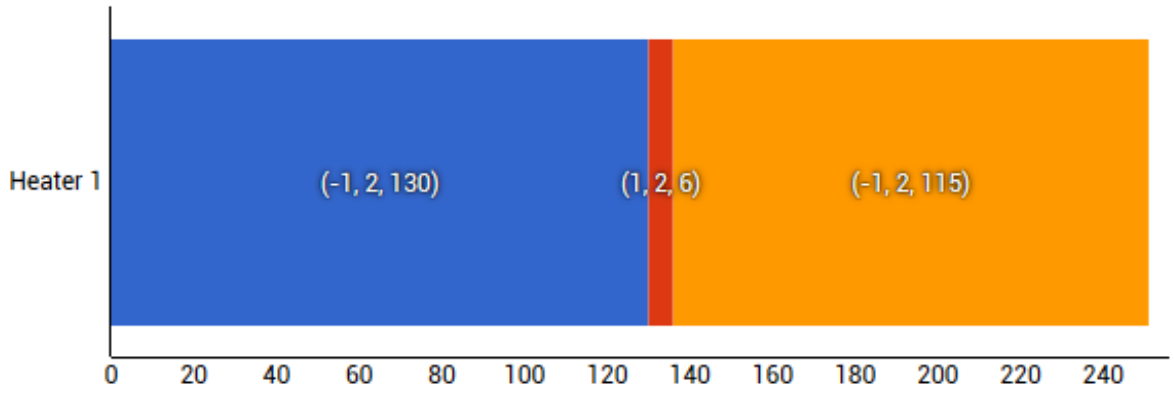


Figura 3.9: Solución del Caso 15 dada por el modelo

Caso 16

Descripción: Utilizando como referencia la Tabla 3.12 se pretende verificar que el modelo es capaz de distribuir la carga del molde $m2$ entre los 7 disponibles. Como solo hay un molde del tipo $m1$ una solución óptima posible es producir en el heater $h1$ $(m1, m2)$ y en los restantes $(m2, m2)$ (ver Figura 3.10). Una alternativa dada por el modelo se puede ver en la Figura 3.11.

Parámetros:

Parámetro	Valor
#heaters	4 (h1, h2, h3, h4)
#moldes	2 (m1 y m2)
#moldes disponibles	m1=1 y m2=7
#demanda	dm1=50u, dm2=500u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.12: Parámetros del Caso 16

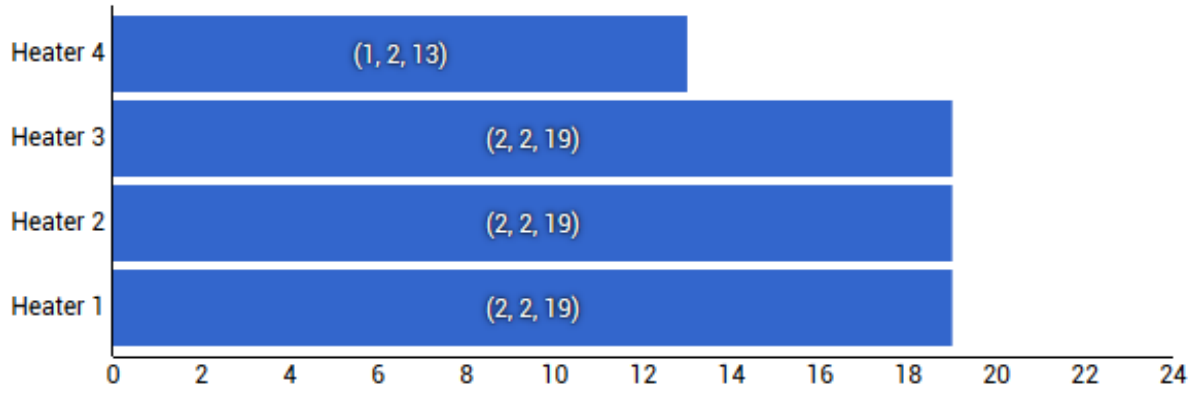


Figura 3.10: Posible solución del Caso 16

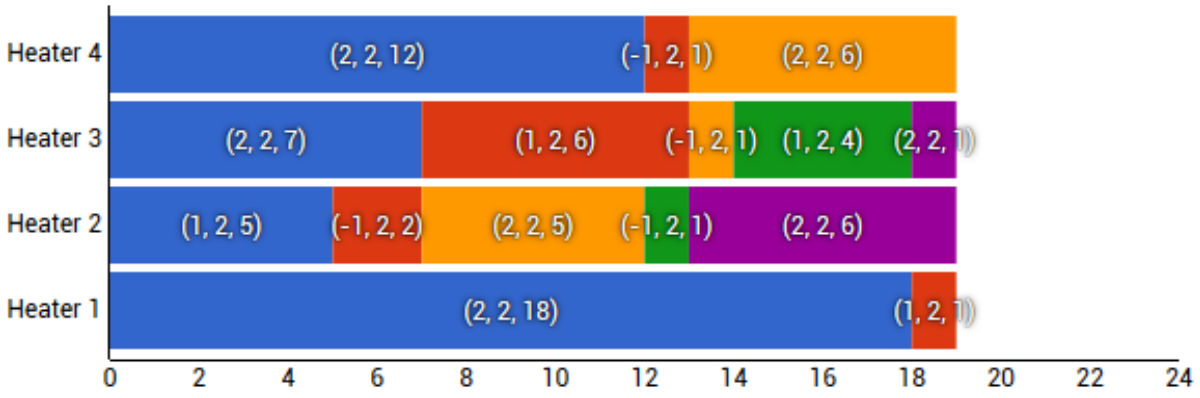


Figura 3.11: Solución equivalente del Caso 16 dada por el modelo.

Caso 17

Descripción: Se busca extender el caso anterior adicionando más moldes disponibles de cada tipo. En estos escenarios donde hay disponibilidad de moldes es ventajoso aprovecha los más posible las asignaciones del tipo (x, x) (producir dos moldes del mismo tipo en un heater) e ir adelantando la producción de otro molde, osea producir en lo posible la asignación (x, y). Aplicándolo a los parámetros de la Tabla 3.13 una posible solución óptima es producir durante 4 periodos las asignaciones $(m1, m1)$, $(m1, m2)$, $(m2, m2)$, $(m2, m2)$ en los heaters $h1, h2, h3$ y $h4$ respectivamente (ver Figura 3.12). El modelo por otro lado genera una solución mejor respecto al exceso de producción del molde $m1$, ver Figura 3.13.

Parámetros:

Parámetro	Valor
#heaters	4 (h1, h2, h3, h4)
#moldes	2 (m1, m2)
#moldes disponibles	m1=3, m2=5
#demanda	dm1=50u, dm2=70u
piezas compartidas	no
tiempo de vulcanizado	tv1=10u, tv2=15u

Tabla 3.13: Parámetros del Caso 17

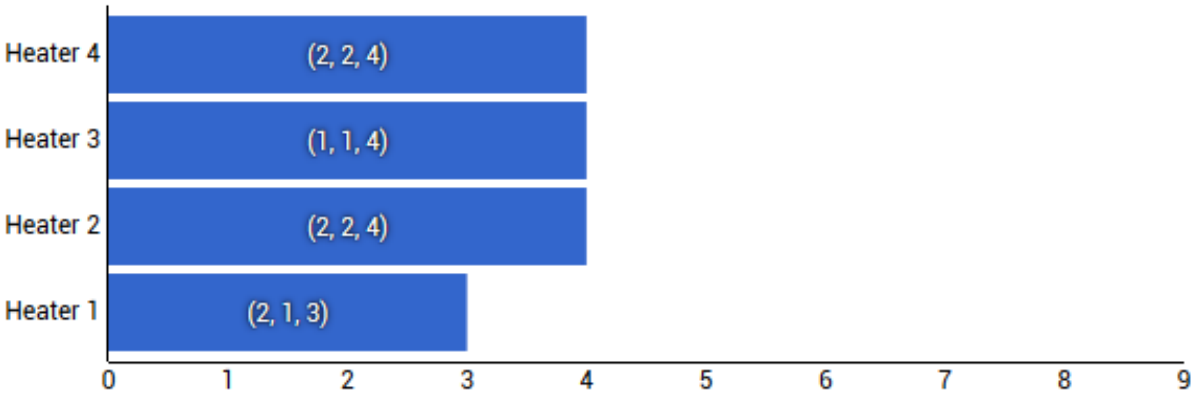


Figura 3.12: Posible solución del Caso 17

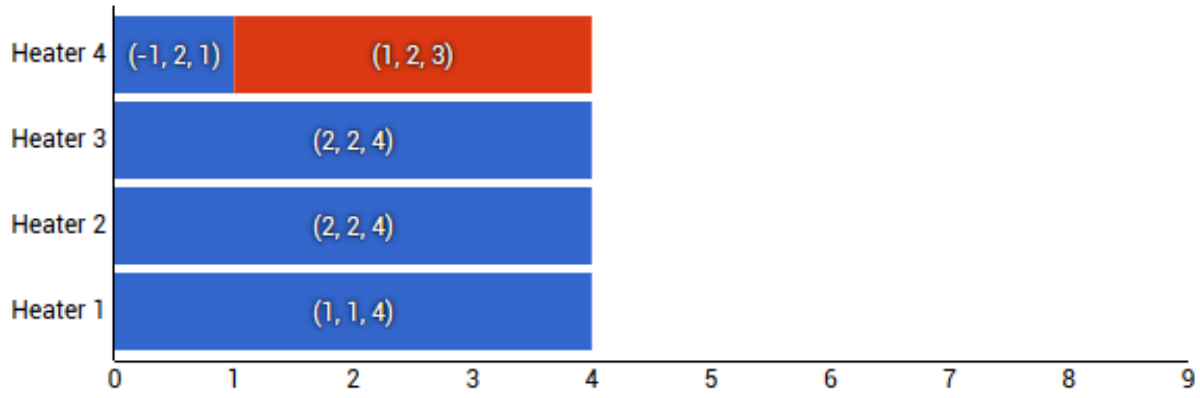


Figura 3.13: Solución equivalente del Caso 17 dada por el modelo.

Caso 18

Descripción: Este caso se enfoca en verificar que la dependencias de piezas que un molde posee no influye en la decisiones de asignación de otros moldes. Para esto la instancia planteada (ver Tabla 3.14) asigna al molde $m3$ como el único que depende de piezas compartidas, por lo tanto éste no debe intervenir en el atraso de la producción de otro molde. Una solución posible es producir durante 2 periodos la asignación $(m1, m1)$ y $(m2, m3)$ durante 4 periodos (ver Figura 3.14); una alternativa dada por el modelo es la que se muestra en la Figura 3.15.

Parámetros:

Parámetro	Valor
#heaters	2 (h1, h2)
#moldes	3 (m1, m2, m3)
#moldes disponibles	m1=3, m2=1, m3=1
#demanda	dm1=20u, dm2=20u, dm3=20
tiempos de vulcanizado	tv1=10, tv2=10, tv3=10
piezas compartidas	si (p1, p2, p3)
#piezas compartidas	p1=1, p2=1, p3 = 1
requerimiento de piezas	m3(p1, p2, p3)

Tabla 3.14: Parámetros del Caso 18

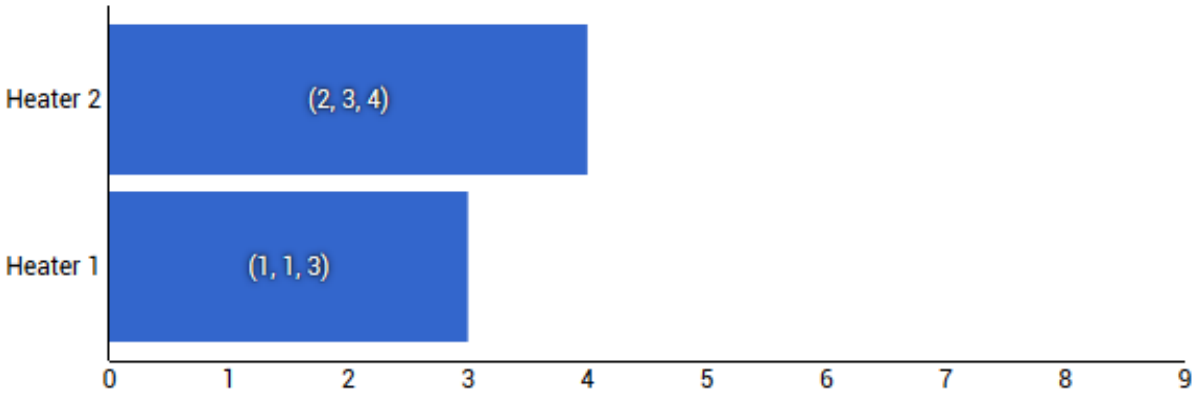


Figura 3.14: Posible solución del Caso 18

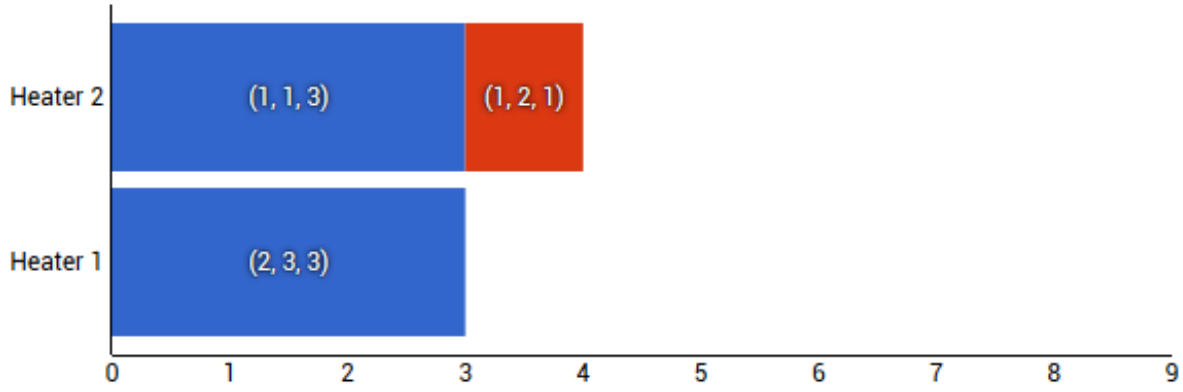


Figura 3.15: Solución equivalente del Caso 18 dada por el modelo

Caso 19

Descripción: El objetivo de este caso es verificar la capacidad del modelo en escenarios más complejos respecto al uso de piezas compartidas. Tomando como referencias la Tabla 3.15 se cuenta con cuatro tipos de pieza, donde además cada molde depende de dos de ellas. Ante estos escenario el modelo debe agrupar un par de moldes donde la intersección del conjunto de piezas requeridas por cada uno sea vacía y además haya disponibilidad de moldes.

Una solución esperada es producir en un heater el par $(m1, m2)$ hasta satisfacer la demanda de ambos moldes y luego producir el par $(m3, m4)$, en la Figura 3.16 se puede ver que se requieren 8 periodos. La Figura 3.17 muestra una solución equivalente dada por el modelo.

Parámetros:

Parámetro	Valor
#heaters	2 (h1, h2)
#moldes	4 (m1, m2, m3, m4)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1
#demanda	dm1=20u, dm2=20u, dm3=20u, dm4=20u
tiempos de vulcanizado	tv1=10, tv2=10, tv3=10
piezas compartidas	sí (p1, p2, p3, p4)
#piezas compartidas	p1=1, p2=1, p3=1, p4=1
requerimientos de piezas	m1(p1, p3), m2(p2, p4), m3(p1, p3), m4(p2, p4)

Tabla 3.15: Parámetros del Caso 19

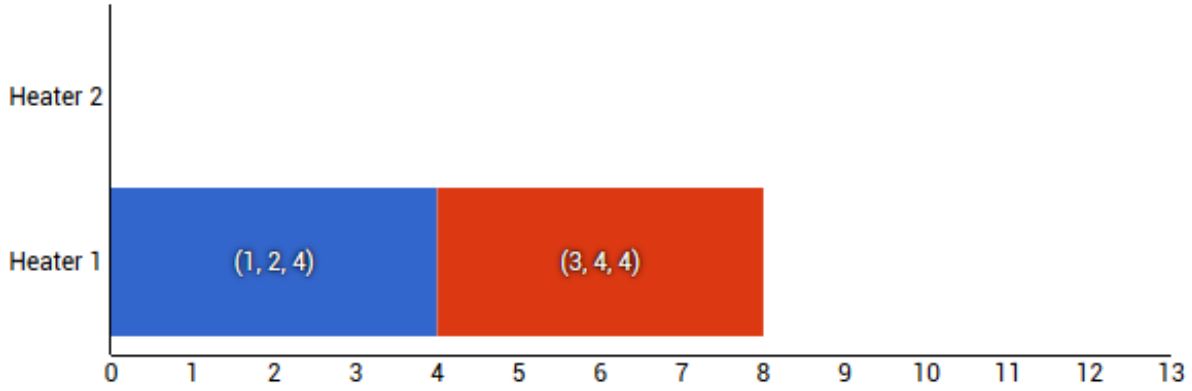


Figura 3.16: Posible solución del Caso 19

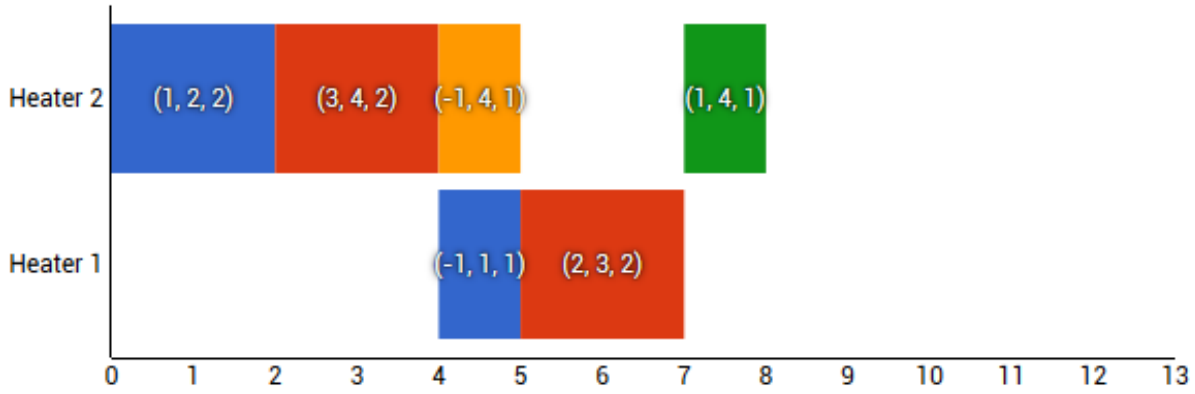


Figura 3.17: Solución equivalente del Caso 19 dada por el modelo

Caso 20

Descripción: Hasta ahora se ha asumido que todos los moldes son compatibles entre sí; en este caso el objetivo es validar las instancias donde existen moldes que no son compatibles entre sí, no se pueden procesar en un mismo heater.

Dada una instancia I con moldes incompatibles entre sí, el modelo debe generar sub-instancias $\{I_1, .., I_n\}$, en donde cada una de ellas los moldes son compatibles entre sí y la cantidad de periodos necesarios es el máximo requerido por I_i. La solución no es así de directa cuando se incluyen en la configuración piezas compartidas.

Tomando como referencia la Tabla 3.16, se generan dos sub-instancias que agrupa los moldes $\{m1, m2\}$ y $\{m3, m4\}$. Para producir la primer asignación se requieren 7 periodos y la segunda 4, por lo tanto la solución global requiere de 7 periodos.

Parámetros:

Parámetro	Valor
#heaters	2 (h1, h2)
#moldes	4 (m1, m2, m3, m4)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1
#demanda	dm1=20u, dm2=37u, dm3=20u, dm4=20u
tiempos de vulcanizado	tv1=10, tv2=10, tv3=10, tv4=10
compatibilidad entre moldes	{m1,m2}, {m3, m4}
piezas compartidas	no

Tabla 3.16: Parámetros del Caso 20

El plan de pruebas propuesto se compone de 14 casos básicos y 6 especiales, el cual fueron evaluados manualmente y comparados con la salida dada por el solver CPLEX. A pesar de que no se incluyen escenarios reales, el objetivo fue analizar y probar posibles patrones que forman escenarios más complejos. A lo largo de las pruebas fue posible encontrar errores en la formulación. Un ejemplo fue el mal cálculo de la producción cuando hay una asignación con moldes del mismo tipo (Casos [5,6]). También se encontraron otros errores de cálculo, como el del tiempo disponible para producir moldes (Casos [7,8]), producción excesiva de moldes (Caso [15]). Además se dieron casos donde el modelo encontró mejor solución a la evaluación manual. Además se realizaron revisiones de instancias más complejas y compuestas por los casos simples definidos en este capítulo. También fue posible corroborar soluciones más complejas

CAPÍTULO 4

Procedimientos de resolución propuestos

4.1. Introducción

En el Capítulo 2 se presentó la formulación matemática que busca un ordenamiento que minimice el tiempo total de producción de neumáticos (makespan). Este capítulo describe el desarrollo de tres procedimientos para resolver esta formulación matemática. Cada uno de ellos aplica técnicas exactas, heurísticas e híbridas respectivamente.

El procedimiento exacto referenciado como "CPLEX" se elaboró codificando la formulación matemática al lenguaje AMPL (A Mathematical Programming Language) que por medio del solver comercial CPLEX es posible resolverlo. Esta manera fue la desarrollada en el Capítulo 3 para validar la formulación matemática. A través del software CPLEX solo se aplican técnicas exactas. En el Capítulo 2 se vio que es necesario estimar el parámetro THB encargado de definir el máximo tiempo disponible para cumplir toda la demanda. Esta estimación se hizo por medio de Ecuación 4.1 el cual es utilizada como una cota superior de la solución óptima. Se verá que este estimador es bueno solo para instancias pequeñas del problema y las desventajas que conlleva.

Ante los inconvenientes que presenta el procedimiento anterior se decidió elaborar un segundo procedimiento que explore técnicas heurísticas. En una primer instancia se elaboró un framework inspirado en la meta-heurística GRASP que expone entidades y funciones genéricas que según su configuración es posible resolver problemas con determinada estructura. Para buscar soluciones este framework se compone de dos fases. La primera es una fase constructiva compuesta por heurísticas constructivas, la segunda fase nombrada como fase de mejora es opcional y se encarga de mejorar cada solución dada por la primer fase. El framework se encarga de generar soluciones a través de las heurísticas constructivas que cuenta, y en caso de estar disponible se le aplica un proceso de mejora.

Para enfrentar el problema de FunsaCoop se utilizó el framework elaborado. Para esto se desarrollaron un conjunto de heurísticas que analizan características específicas del problema. Por medio de ellas se obtienen soluciones aproximadas a las óptimas y que requieran bajo tiempo de resolución. El algoritmo final se referencia como "Estimador". La elaboración de este algoritmo incluye un trabajo de validación, el cual se validan las estructuras de las soluciones generadas por el Estimador. Los casos de pruebas definidos en el Capítulos 3 son reutilizados. De esta forma se corroborará que las soluciones cumplan con todas las restricciones del problema. No se dará importancia a la exactitud de las soluciones ya que se está aplicando técnicas de aproximación. Este capítulo también describe en detalles aspectos de arquitectura e implementación aplicados para el desarrollo del framework. Además se explica como las entidades dadas por este fueron aplicadas para obtener el Estimador que resuelve efectivamente el problema de vulcanizado de FunsaCoop. Finalmente se expone una interfaz gráfica web desarrollada para facilitar la instanciación del problema, carga, visualización de datos y configuración del Estimador.

Por último, el tercer procedimiento de resolución combina lo mejor de los procedimientos anteriores. Esta metodología aplica técnicas exactas y heurísticas. Para mejorar la situación del procedimiento exacto se utiliza Estimador como forma de calcular el parámetro THB. Con un valor razonable es posible reducir la cantidad de variables y restricciones que el solver CPLEX debe manejar. Como resultado se obtuvieron mejoras en tiempos de resolución y calidad de soluciones al momento de resolver la formulación matemática. Este procedimiento se

referencia como “CPLEX+Estimador”

Este capítulo presenta en la Sección 4.2 los procedimientos desarrollados, en la Sección 4.3 la validación correspondiente al Estimador, en la Sección 4.4 se describe la arquitectura del framework elaborado, en la Sección 4.5 las entidades desarrolladas para resolver el problema y por último en la Sección 4.6 la interfaz gráfica para poder ejecutar al Estimador.

4.2. Procedimientos propuestos

4.2.1. Método exacto (CPLEX)

Este primer procedimiento fue elaborado en el Capítulo 2 al codificar la formulación matemática al lenguaje de modelado matemático AMPL. De las distintas herramientas que existen para resolver modelos AMPL se decidió utilizar CPLEX por su potencial de resolución que ofrece. Existen otras como GLPK, Gurobi, Xpress usados para los mismo fines. CPLEX tiene la capacidad de analizar y aplicar procedimientos al modelo AMPL para tratar de reducir las cantidad de variables y restricciones. Además explora distintas técnicas exactas para determinar cual se ajusta mejor para resolver el modelo. A través de CPLEX es posible obtener soluciones óptimas para el problema de FunsaCoop. Esto quiere decir que no existe un ordenamiento de la producción que requiera menos tiempo para producir toda la demanda de neumáticos.

La formulación matemática presentada en el Capítulo 2 requiere del parámetro THB, cuyo valor determina la cantidad de variables y restricciones aplicadas. Por este motivo debe tener un valor razonable para no generar instancias del modelo más grandes de lo necesario. La manera de resolver este problema es estimar este parámetro. La Ecuación 4.1 es un estimador del parámetro THB cuyo valor está determinado por los demás parámetros de la formulación matemática. Tal ecuación representa la cantidad de periodos necesarios para producir toda la demanda si solo se cuenta con un heater y un solo molde para cada tipo de neumático. El valor estimado corresponde a disponer la producción de moldes en forma secuencial. El primer factor representa la cantidad de neumáticos que se pueden producir en un periodo de tiempo cuando se le sustrae el máximo tiempo de colocado y quitado divido el máximo tiempo de vulcanizado. Los dos últimos factores indica la demanda total de neumáticos utilizando el valor máximo demandado por la cantidad de moldes.

$$THB = \lceil \frac{\phi - max\{tc_i | i \in M\} - max\{tq_i | i \in M\}}{max\{tv_{i,k} | (i,k) \in M \times H\}} * |M| * max\{dm_i | i \in M\} \rceil \tag{4.1}$$

En escenarios donde halla muchos moldes o la demanda sea grande el valor obtenido puede ser una cota superior muy alejada al valor óptimo ya que en la ecuación se utiliza el máximo valor de demanda. Además en muchas instancias del problema se cuanta con más de un heater y muchos de los moldes se pueden producir en simultaneo. No obstante CPLEX analiza la estructura del modelo para reducir la cantidad de variables y restricciones, además explora distintas técnicas de resolución para disminuir el tiempo de ejecución. Pese al intento existen casos donde CPLEX no es capaz de resolver en un tiempo razonable escenarios reales del problema, algunas invocaciones exceden las 8 horas de ejecución, incluso existen escenarios que no es capaz de resolver. Esto es una característica propia de los métodos exactos, su análisis exhaustivo requiere más procesamiento y tiempo. Otro inconveniente presentado es la representación de soluciones que genera CPLEX. En distintos escenarios se obtienen distintas representaciones, lo cual implica realizar un esfuerzo mayor para analizarla y generar una representación más fácil de visualizar. Además para utilizar CPLEX es necesaria la compra de licencias, lo cual conlleva un gasto adicional.

4.2.2. Algoritmo Estimador

Visto lo anterior se decide explorar otras alternativas para resolver el problema de FunsaCoop. En el Anexo B se examinaron varias formas de resolver problemas combinatorios, el cual se pueden separar en dos clases: técnicas exactas y de aproximación. Este apartado se inclina en las técnicas de aproximación, en especial a las heurísticas dado que sus aplicaciones han dado buenos resultados en problemas combinatorios.

En esta rama se encuentra una amplia variedad de heurísticas [1] tales como Algoritmos Evolutivos (GA), Búsqueda Tabú (TS), Greedy Randomized Adaptive Search Procedure (GRASP), Particle Swarm Optimization (PSO), Simulated Annealing (SA), etc. Muchas de estas heurísticas se basan en fenómenos biológicos, químicos y sociales como forma de explorar el espacio de búsqueda y llegar a una solución óptima o aproximada. La velocidad de resolución que otorgan estas técnicas son una de sus principales características, pero implica sacrificar calidad de solución.

La segunda forma de tratar el problema de FunsaCoop está basada en la forma que funciona la meta-heurística GRASP. Se eligió porque tiene una estructura simple el cual lo hace rápido y fácil de desarrollar, es iterativo y se compone por dos fases. La primera es una fase constructiva (FC) mientras que la segunda es de mejora (FM). Lo interesante es que en la primer fase se puede aplicar una variedad de heurísticas para mejorar la exploración del espacio de búsqueda, haciéndolo un algoritmo extensible. Otras heurísticas como GA o PSO son un poco más complejas y requieren ser calibradas y configuradas para que funcionen como se espera. Adicionando el tiempo de entrega del proyecto GRASP se adecua mejor.

La estructura básica de GRASP se muestra en el Algoritmo 1. Esta heurística genera tantas soluciones candidatas como iteraciones se le habiliten. En cada iteración se genera una solución candidata, luego se aplica una búsqueda local y se compara con la mejor solución actualmente encontrada. La generación de soluciones representa la FC, mientra que la búsqueda local representa la FM. Las lineas [4, 5] representan estas fases.

Framework

Como se pretende desarrollar una herramienta extensible en el proceso de búsqueda se decidió elaborar un framework. La idea es ofrecer una herramienta que por medio de entidades y funciones genéricas sea posible resolver problemas con una estructura determinada. El framework desarrollado está compuesto por un algoritmo inspirado en la meta-heurística GRASP y un conjunto de entidades sobre las que trabaja. Cada entidad debe tener definida ciertas propiedades que son específicas al problema a resolver.

El algoritmo genérico se compone por una Fase Constructiva (FC) y una Fase de Mejora (FM). La FC está parametrizada por un conjunto de heurísticas constructivas encargadas de generar soluciones del problema a enfrentar. La FM es opcional y está destinada a ser invocada por cada solución generada en FC con fines de mejorarlas. El framework se encarga de generar las soluciones a través de las heurísticas tantas como estas lo indiquen y en caso de desponer de una FM se aplica el proceso de mejora a cada solución generada en FC. El framework puede utilizarse en problemas que puedan separarse en dos fases (FC y FM).

Aquí surgen varias ventajas y desventajas. Como ventajas se tiene 1) la capacidad de resolver problemas que puedan separarse en dos fases, 2) por medio de las heurísticas constructivas se puede ampliar la exploración en el espacio de búsqueda, 3) como cada una de ellas no comparten información se pueden aplicar técnicas para generar soluciones en simultaneo permitiendo reducir tiempos de resolución y 4) adicionando más heurísticas es posible seguir aumentando la capacidad de búsqueda de soluciones. Como desventajas se tiene que 1) cuando más heurísticas constructivas se agreguen puede tener efectos negativos en el tiempo de resolución y la 2) complejidad y análisis que apliquen puede jugar en contra.

A continuación se explican las etapas de elaboración del algoritmo genérico contenido en el framework.

La meta-heurística GRASP (Algoritmo 1) se compone por una fases principales, una de construcción de soluciones (fase constructiva) y otra que aplica Búsqueda Local (fase de mejora). La FC se generalizó para soportar varias heurísticas que ayuden explorar el espacio de búsqueda. La FM correspondiente a la Búsqueda Local en GRASP se generalizó para ser un proceso de mejora y opcional. El resultado obtenido es el Algoritmo 2. Como se dijo anteriormente el framework genera las soluciones a través de las heurísticas constructivas y explora mejoras por medio del proceso de mejora. Cada heurística constructiva brinda un constructor de soluciones a través de la función “generarSolucion” y una propiedad “maxIteraciones” que indica cuantas soluciones es capaz de generar. Esto se puede ver en el procedimiento 10. Asimismo deben ser capaz de generar al menos una solución.

Las heurísticas constructivas se pueden clasificar como determinista o aleatoria. Para una instancia del problema, una heurística determinista genera siempre la misma solución (la propiedad “maxIteraciones” debe valer 1), mientras que una aleatoria tiene la capacidad de generar más de una solución (la propiedad “maxIteraciones”

[1] Según Zanakins y Evans (1981), “una heurística es un procedimiento simple, a menudo basado en el sentido común, que se supone que ofrecerá una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido”.”

puede ser mayor o igual a 1). La FM en GRASP corresponde a realizar una búsqueda local, en el algoritmo genérico se generaliza a un proceso opcional de mejora de soluciones. En caso de que esté disponible se aplica el proceso a cada una de las soluciones generadas por las heurísticas constructivas. Este proceso puede desarrollarse como un procedimiento de búsqueda local o un medio para corregir soluciones.

Con el framework es posible construir distintos tipos de heurísticas para resolver problemas combinatorios. Por ejemplo, si se utiliza una heurística constructiva aleatoria y un proceso de Búsqueda Local se obtiene efectivamente el algoritmo GRASP. De igual forma es posible obtener la heurística Tabú Search y Ant Colony Optimization ya que siguen una estructura similar a la definida para el framework, se componen por una FC y FM.

Como GRASP no mantiene información salvo la solución global es posible aislar la tarea de construcción y mejora de una solución. Esto permite aprovechar la arquitectura de las computadoras modernas, dejando así construir soluciones en simultaneo y reducir el tiempo ejecución. Para lograrlo el Algoritmo 2 se modificó obteniendo así el Algoritmo 3. En el procedimiento principal del Algoritmo 3 se ejecuta cada una de las heurísticas constructivas a través del procedimiento "EjecutarHeuristica", el cual internamente cada solución se construye en una "tarea" capaz de correr en simultaneo (procedimiento "crearTarea" 32). Posteriormente se espera a que todas las tareas hallan finalizado para comparar la soluciones generadas (linea 13).

La versión final del framework sigue la estructura del Algoritmo 3, otorgando la capacidad de agregar procedimientos heurísticos constructivos, complementados por un proceso de mejora y capaz de explorar el espacio de búsqueda a través de la generación de soluciones en simultaneo. Las siguientes secciones detallan aspectos de arquitectura, como las entidades y funciones necesarias definir para resolver problemas específicos.

Estimador

Para resolver el problema de FunsaCoop se utilizó el framework elaborado. El trabajo consistió en definir que heurísticas constructivas elaborar y como se debería hacer el proceso de mejora para encontrar mejores soluciones. Como resultado se propusieron tres heurísticas constructivas capaces de analizar patrones de planificación específicos al problema de FunsaCoop. Cada una tiene su forma de generar una solución. Se incluye además un proceso de mejora para ajustar y corregir ciertos patrones en las soluciones generadas. De ahora en adelante el uso del framework junto a las heurísticas para resolver el problema de vulcanizado se referencia como "Estimador". Esta metodología permitió hacer heurísticas sencillas con poca demanda de cómputos. Además el análisis no exhaustivo y el uso de estructuras de datos eficientes son una forma de obtener mayor velocidad de resolución. Las heurísticas constructivas y el proceso de mejora elaborados para el problema de FunsaCoop se detallan a continuación.

Algoritmo 1 GRASP

```
1:  procedimiento Principal
2:      solucionGlobal ← ∅
3:      para cada 1...maxIteraciones hacer
4:          solucionLocal ← construirSolucion()
5:          solucionLocalMejorada ← busquedaLocal(solucionLocal)
6:          sí solucionLocalMejorada mejor que solucionGlobal entonces
7:              solucionGlobal ← solucionLocalMejorada
8:          fin
9:      fin
10:     retornar solucionGlobal
11: fin
```

Fase constructiva Visto lo anterior se procede a explicar los componentes desarrollados para resolver el problema de FunsaCoop a través del Estimador. La FC del Estimador la representa la función "generarSolucion" de una heurística constructiva (Algoritmo 3, linea 33) y se ocupa de construir un ordenamiento de la producción para el problema de FunsaCoop.

Como se vio en los Anexos [A, B] la representación de datos en las heurísticas juegan un rol importante dado que interviene en todo el proceso de búsqueda. Las estructuras de datos utilizadas influyen en el rendimiento de los algoritmos al momento de ser construidas y consultadas. Para este problema la representación de un

Algoritmo 2 Framework, Algoritmo Genérico (primer versión)

```
1: procedimiento Principal
2:     heuristicasConstructivas ← {h_1...h_n}
3:     solucionGlobal ← ∅
4:     para cada heuristica ← heuristicasConstructivas hacer
5:         solucionGlobal ← EjecutarHeuristica(heuristica, solucionGlobal)
6:     fin
7:     retornar solucionGlobal
8: fin
9:
10: procedimiento EjecutarHeuristica(heuristica, solucionGlobal)
11:     maxIteraciones ← heuristica.maxIteraciones()
12:     para cada 1...maxIteraciones hacer
13:         solucionLocal ← heuristica.generarSolution()
14:         solucionLocalMejorada ← mejorarSolucion(solucionLocal)
15:         sí solucionLocal mejor que solucionGlobal entonces
16:             solucionGlobal ← solucionLocal
17:         fin
18:     fin
19:     retornar solucionGlobal
20: fin
```

ordenamiento queda definida por un conjunto de heaters el cual contienen un conjunto de asignaciones ordenados por fecha de producción. Cada asignación mantiene información de que tipo de moldes hay que producir, la cantidad a producir y los periodos de inicio y fin asignados. Esta representación tiene como ventaja ser simple y fácil de construir. Sin embargo mantener los conjuntos de asignaciones ordenados puede ser costoso. Como se verá, la forma de construir esta representación no genera este problema.

En las visitas realizadas a FunsaCoop los trabajadores resaltaron que es preferible producir una asignación de moldes en un heater lo más posible para ahorrar tiempo en re configurar máquinas y evitar tener la caldera en estado ocioso. Para cumplir este requerimiento la función "generarSolucion" de una heurística se compone por dos etapas. La primera se encarga de generar asignaciones seleccionando moldes compatibles y la segunda de distribuir estas asignaciones en heaters compatibles. Juntas generan un ordenamiento. Cada heurística constructiva propuesta en este proyecto sigue una estructura básica pero difieren en como seleccionan pares de moldes.

Entrando en detalles la primer etapa se encarga de 1) particionar la demanda de cada neumático. El objetivo es distribuir la demanda entre la cantidad de moldes disponibles para poder repartir el trabajo entre los heaters. Por ejemplo, si del tipo de neumático X se precisan 300 unidades y además se cuenta con 3 moldes que lo pueden moldear, la partición es de 3 fragmentos de 100 unidades. Posteriormente 2) se seleccionan pares de moldes compatibles para generar una asignación. En este proyecto se proponen tres formas de agrupar moldes, cada una define una heurística constructiva nombradas como HCTV, HCA y HCD.

Comenzando con HCTV (Heurística Constructiva por Tiempo de Vulcanizado), esta tiene como política seleccionar moldes compatibles con menor diferencia en tiempo de vulcanizado. Ya que la cantidad producida de neumáticos en una asignación se limita al mayor tiempo de vulcanizado, cuanto menor sea esta diferencia mayor va a ser la producción de ambos moldes y así es posible reducir el tiempo de producción. Una característica importante es que este proceso es determinista, esto quiere decir que para una instancia del problema la solución generada va a ser la misma independiente de la propiedad "*maxIteraciones*".

Por otro lado HCA (Heurística Constructiva Aleatoria) aplica la política de seleccionar dos moldes compatibles de forma aleatoria. La finalidad es expandir el espacio de búsqueda y no estancarse en una solución candidata. En pruebas realizadas este proceso en promedio mejora notoriamente la calidad de solución por la diversidad de soluciones que genera. En este caso la propiedad "*maxIteraciones*" puede ser mayor a uno dado que se pueden generar más de una solución. Otro punto a favor es su sencillez como rutina, ya que no aplica ningún análisis complejo, solo se encarga de seleccionar aleatoriamente un par de moldes.

Por último HCD (Heurística Constructiva por Demanda) es otro proceso determinista donde la política de agrupación se basa en seleccionar moldes compatibles con menor diferencia en demanda. Este tipo de agrupación

Algoritmo 3 Framework, Algoritmo Genérico (versión final)

```
1:  procedimiento Principal
2:      heuristicasConstructivas ← {h_1...h_n}
3:      solucionGlobal ← ∅
4:      tareas ← ∅
5:
6:      para cada heuristica ← heuristicasConstructivas hacer
7:          tareas ← tareas ∪ EjecutarHeuristica(heuristica)
8:      fin
9:
10:     Esperar que finalicen tareas
11:
12:     para cada tarea ← tareas hacer
13:         sí tarea.solucion es mejor que solucionGlobal entonces
14:             solucionGlobal ← tarea.solucion
15:         fin
16:     fin
17:     retornar solucionGlobal
18: fin
19:
20: procedimiento EjecutarHeuristica(heuristica)
21:     tareas ← ∅
22:     maxIteraciones ← heuristica.maxIteraciones()
23:
24:     para cada 1...maxIteraciones hacer
25:         nuevaTarea ← crearTarea(heuristica)
26:         tareas ← tareas ∪ nuevaTarea
27:         ejecutar(nuevaTarea)
28:     fin
29:     retornar tareas
30: fin
31:
32: procedimiento crearTarea(heuristica)
33:     solucionLocal ← heuristica.generarSolution()
34:     solucionLocalMejorada ← mejorarSolucion(solucionLocal)
35:     retornar solucionLocalMejorada
36: fin
```

genera menos intercambios de moldes ya que la producción culmina casi al mismo tiempo, por lo tanto se evita configurar heaters de forma innecesaria, aumentando la producción y así reduciendo el tiempo total.

Las tres formas se comportan de diferente manera según la instancia del problema. Por ejemplo en instancias donde hay disponible varios moldes del mismo tipo el proceso HCTV tiende a comportarse mejor que los otros. Las 3 modalidades son rutinas sencillas y rápidas, esto se logró porque las formas de agrupar moldes analizan escenarios con características especificas.

Contando con las demandas particionada y las asignaciones se procede a 3) asignar la cantidad de neumáticos a producir en cada asignación. La manera de hacerlo es elegir la menor fracción de la demanda de los neumáticos que componen una asignación. Por ejemplo, si una de ellas se compone por los moldes M y N y su fracciones son 100 y 150 respectivamente, la cantidad a producir de cada molde va a ser 100, los 50 restantes del molde N se acumularán para futuras asignaciones.

Por último 4) se retiran los moldes del conjunto de moldes disponibles que satisfacen su demanda para evitar exceso de producción, en la medida que se generan asignaciones los moldes removidos no serán utilizados.

Estos 4 pasos nombrados se repiten hasta que la demanda de cada neumático se satisfaga. En el Algoritmo 4.2.2 se muestra el pseudocódigo. Las lineas 9, 14, 15, 20 representan las etapas 1, 2, 3 y 4 respectivamente. Como resultado de esta primer etapa se obtiene un conjunto de asignaciones con las cantidades asignadas de neumáticos a producir para así respetar los requerimientos de FunsaCoop, además se contemplan las restricciones de compatibilidad entre moldes y demanda de neumáticos.

La segunda etapa utiliza el conjunto de asignaciones generadas anteriormente y se encarga de distribuirlas adecuadamente en los heaters por medio de un proceso de apilado. Para esto se selecciona una asignación y se agrega a un heater compatible que posea la pila con menor periodos utilizados.

Para determinar que asignación es la más adecuada se aplican tres consultas sobre el conjunto de asignaciones. Cada consulta devuelve el menor periodo de disponibilidad. La primer consulta determina el menor periodo que un heater compatible está disponible, la segunda determina el menor periodo en función de la disponibilidad de moldes y la tercera determina el menor periodo disponible en función de la disponibilidad de piezas compartidas. Para cada asignación se toma el máximo valor de las consultas para cumplir con las restricciones del problema. La asignación que obtuvo el menor valor es la seleccionada. Este proceso se repite tantas veces como asignaciones haya para ubicar. Aquí se chequean las restricciones de disponibilidad de moldes, piezas compartidas, compatibilidad molde-heater y las basadas en la relación de precedencia entre asignaciones. En el Algoritmo 4.2.2, linea 26 se puede ver el pseudocódigo de esta segunda etapa.

Fase de mejora La FM de Estimador (correspondiente a la función "*mejorarSolucion*" en el Algoritmo 3) tiene como objetivo tratar de generar una mejor solución partiendo de la solución dada en la FC. En la literatura revisada muchos trabajos aplican búsqueda local, el cual consiste en empezar por una solución inicial y buscar en su vecindad una mejor solución; si la encuentra, reemplaza la solución actual por la nueva y continua con el proceso hasta que no se pueda mejorar la solución actual. Por otro lado se puede aplicar una rutina de mejora, que consiste en revisar patrones y corregirlos para dar lugar a una mejor solución.

En observaciones realizadas en la FC se notaron ciertos patrones que dan lugar a posibles correcciones para reducir el tiempo total de producción. Se decidió realizar estas correcciones en la FM del Estimador ya que se puede reutilizar en todas las heurísticas constructivas. Como se hallaron dos posibles mejoras este proceso se separa en dos pasos.

El primero se vincula al resultado del proceso de apilado. Algunas pilas más largas que otras pueden ser particionada y distribuidas en otros heaters compatibles y a la vez seguir satisfaciendo todas las restricciones. El objetivo es tratar de nivelar las pilas para lograr una reducción del tiempo de producción. Para esto se elije aleatoriamente un subconjunto pequeño de asignaciones que tenga dos moldes asignados y de distintos tipos para luego aplicar una de dos posibles acciones. Una de las acciones es separar la asignación en dos asignaciones manteniendo la cantidad a producir pero dejando un compartimiento vacío en cada una. Esta decisión se basa en que la cantidad de neumáticos producidos en una asignación la limita el tiempo de vulcanizado de los tipos de moldes que alberga. Separarlos puede generar más neumáticos producidos de un tipo y terminar su producción más temprano. La otra acción consiste en generar dos asignaciones manteniendo los moldes pero asignando 1/3 a una y 2/3 a la otra. Esta decisión busca equilibrar de forma precavida la carga en los heaters compatibles, cuanto más nivelados estén es posible reducir el tiempo total de producción.

El siguiente paso aplicado se vincula a como las asignaciones de un heater se relacionan. Si dos asignaciones consecutivas mantienen algún molde en común entonces se evita parte de la configuración del heater al pasar de una asignación a otra, en consecuencia ese tiempo se compensa en producir más neumáticos. Si esto se aplica varias veces se puede ocasionar exceso de producción. Este paso trata de recortar lo producido en exceso trabajando con las asignaciones ubicadas en la cima de cada pila y se aplica mientras las restricciones de demanda para cada neumático se cumplan.

Estas rutinas modifican una solución para tratar de obtener una que requiera menos tiempo de producción, pero también es posible generar una peor a la original. En caso de que sea mala la solución generada se mantiene la original. El resultado de estos pasos da un nuevo conjunto de asignaciones el cual luego es procesado por el procedimiento "distribuirEnHeaters" del Algoritmo 4.2.2. Ambos pasos son sencillos y eficientes ya que no es necesario un análisis exhaustivo de toda la solución, por este lado es bueno porque no genera alto costo computacional. Sin embargo tiene como desventaja que la calidad en mejora depende de la estructura de la solución. Por ejemplo en el segundo paso los últimos moldes quizás no son los más adecuados para aplicar el recorte.

Ya que los pasos descritos anteriormente pueden dar lugar a más mejoras, se aplican a lo sumo 5 veces para tratar de seguir mejorando la solución. Este proceso de mejora no corresponde a una búsqueda local dado que no se explora toda la vecindad de las soluciones generadas. El Algoritmo 4.2.2 presenta el pseudocódigo de la fase de mejora.

La segunda forma propuesta para resolver el problema de FunsaCoop se inspiró en la estructura de GRASP y como resultado genera un ordenamiento de la producción. El Algoritmo Estimador presentado otorga la posibilidad de combinar varias técnicas heurísticas para ampliar la búsqueda en el espacio de soluciones por medio de una fase constructiva y de mejora. Para su FC se desarrollaron 3 heurísticas constructivas (HCTV, HCA, HCD) el cual definen una política de agrupación de moldes. Separar el análisis de una instancia del problema en varias rutinas permitió construir algoritmos simples y eficientes. En la FM se aplican rutinas que corrigen y ajustan las soluciones generadas por la fase anterior, este proceso es opcional. Además Estimador es capaz de generar soluciones de forma simultanea gracias a la arquitectura de las computadoras modernas permitiendo así reducir más los tiempos de ejecución.

4.2.3. Método híbrido (CPLEX+Estimador)

En el procedimiento exacto donde la formulación matemática se codificó en AMPL, se vio que el tiempo de resolución y tamaño de este se ve afectado por el parámetro THB, lo cual puede ser desventajoso en algunos escenarios. En una primer instancia se planteó la ecuación 4.1 como forma de estimarlo, el cual genera valores grandes (cuando no corresponde) en instancias con muchos moldes o gran demanda. Una tercer versión se desarrolló con el objetivo de mejorar esta situación por medio de técnicas híbridas, el cual consiste en combinar métodos exactos y de aproximación. Varios de los trabajos revisados han aplicado esta metodología con el fin de aprovechar las ventajas que ofrecen los métodos exactos y de aproximación para resolver problemas combinatorios.

La idea es utilizar el algoritmo Estimador dado que genera un ordenamiento de la producción y en consecuencia se sabe cuantos periodos son requeridos. Este valor no siempre es el óptimo pero sirve como un valor aproximado y razonable para utilizarlo en el parámetro THB. De esta manera el procedimiento consiste en calcular un valor del parámetro THB a través de Estimador y utilizarlo luego en la ejecución de CPLEX. En pruebas realizadas esta modalidad ayudó notoriamente al solver para reducir el tiempo de resolución del problema y en consecuencia fue posible obtener más ordenamientos óptimos de la producción. No obstante se llevan las desventajas de utilizar CPLEX: existen instancias donde el solver no es capaz de resolver, el formato de datos devuelto es complejo de visualizar y es necesario comprar licencia de software.

4.2.4. Resumen

En esta sección se presentaron 3 maneras de resolver el problema de FunsaCoop variando la metodologías de resolución. La primera aplica métodos exactos por medio de un software comercial capaz de generar ordenamientos de la producción óptimos. La segunda referenciado como Estimador se basa en técnicas heurísticas e inspirada en GRASP el cual otorga soluciones próximas a las óptimas y velocidad de resolución. La última propuesta combina el potencial de las anteriores.

Algoritmo 4 Fase constructiva

```
1: procedimiento generarSolucion(datos)
2:     asignaciones ← generarAsignaciones(datos)
3:     solucion ← distribuirEnHeaters(asignaciones)
4:     retornar solucion
5: fin
6:
7: procedimiento generarAsignaciones(datos)
8:     moldesDisponibles ← datos.moldes
9:     demanda ← particionarDemanda(datos.demanda)
10:    producido ← ∅                          ▷ Para registrar lo producido de cada molde
11:    asignaciones ← ∅                                ▷ Registro de asignaciones creadas
12:    mientras moldesDisponibles ≠ ∅ hacer
13:
14:        molde1, molde2 ← generarSiguienteAsignacion(moldesDisponibles)
15:        paraProducir ← obtenerDemandaApropiada(molde1, molde2, demanda)
16:
17:        nuevaAsignacion ← (molde1, molde2, paraProducir)
18:        asignaciones ← asignaciones ∪ nuevaAsignacion
19:
20:        actualizarProduccion(producido, datos)
21:        actualizarMoldesDisponibles(moldesDisponibles, producido, datos)
22:    fin
23:    retornar asignaciones
24: fin
25:
26: procedimiento distribuirEnHeaters(asignaciones, datos)
27:    heaters ← {h1...hn}
28:    mientras asignaciones ≠ ∅ hacer
29:        asignacion, periodo, heater ← obtenerMejorAsignacion(asignaciones, heaters)
30:        asignacion.periodo ← periodo
31:        heater.agregar(asignacion)
32:        asignaciones.remover(asignacion)
33:    fin
34:    retornar Solucion(heaters)
35: fin
36:
37: procedimiento obtenerMejorAsignacion(asignaciones, heaters)
38:
39:    minPeriodo ← ∞
40:    mejorAsignacion ← ∅
41:
42:    para cada x ← asignaciones hacer
43:
44:        periodo1 ← obtenerDisponibilidadPorHeater(x, heaters, datos)
45:        periodo2 ← obtenerDisponibilidadPorMoldes(x, heaters, datos)
46:        periodo3 ← obtenerDisponibilidadPorPiezas(x, heaters, datos)
47:        maxPeriod ← max(periodo1, periodo2, periodo3)
48:
49:        sí maxPeriod < minPeriodo entonces
50:            minPeriodo ← maxPeriod
51:            mejorAsignacion ← x
52:        fin
53:    fin
54:    retornar mejorAsignacion
55: fin
```

Algoritmo 5 Fase de mejora

```
1: procedimiento mejorarSolucion(solucion)
2:     mejorSolucion ← ∞
3:     maxIntentos ← 5
4:     idx ← 0
5:     seguirProbando ← 1
6:     mientras idx < maxIntentos y seguirProbando == 1 hacer
7:         solucion_1 ← nivelarPilas(mejorSolucion)
8:         solucionLocal ← ajustarExcesoProducido(solucion_1)
9:         sí solucionLocal es mejor que mejorSolucion entonces
10:            mejorSolucion ← solucionLocal
11:        else
12:            seguirProbando ← 0
13:        fin
14:        idx ← idx + 1
15:    fin
16:    retornar mejorSolucion
17: fin
```

En pruebas realizadas se vieron que en escenarios de tamaño mediano (correspondiente a los reales de Funsa-Coop) no se pueden resolver por medio de métodos exacto debido a la dimensiones del problema. Por otro lado el Estimador es capaz de generar soluciones óptimas en escenarios chicos/medianos a una velocidad mucho mayor.

4.3. Validación

Esta sección describe la etapa de validación del algoritmo "Estimador". El plan de pruebas propuesto en el Capítulo 3 fue reutilizado para validar solo la estructura de las soluciones, debido a que se el algoritmo elaborado aplica técnicas de aproximación. Aquí se utilizaron en el Estimador las tres heurísticas constructivas HCTV, HCA y HCD y el proceso de mejora. Ya que HCA es capaz de generar más de una solución se decidió que genere 500.

Los casos de pruebas utilizados son los definidos en las Secciones 3.3.1 y 3.3.2 del Capítulo 3. Las soluciones óptimas obtenidas en los casos fueron utilizadas como referencia para ver que tan alejadas son las soluciones generadas por el Estimador. Nos obstante la exactitud no fue tomada en cuenta, ya que importa más que las soluciones sean validen las restricciones del problema.

Las pruebas se realizaron utilizando la PC con las características descritas en la Tabla 3.2. A medida que los casos de validación fueron ejecutados surgieron fallas en las soluciones del Estimador, específicamente en los ordenamientos. En el Caso 11 se encontró una falla en la generación del conjunto de asignaciones en la fase constructiva. Se construía una asignación que contenía dos moldes iguales. Esto no era válido ya que existe una pieza compartida por todos los moldes, no se debe tener una asignación con dos moldes. Por otro lado en el Caso 17 surgió una falla en la fase de mejora al momento de ajustar la producción en exceso. Se daban casos donde el recorte se de hacía de forma innecesaria y se generaban soluciones que violan la restricción de demanda. También fue posibles corregir fallas a más bajo nivel.

Una vez corregidas las fallas y validado todos los casos se obtuvo que de los 20 definidos 19 corresponden a soluciones exactas. El caso 6 fue el único que difirió con la solución exacta, sin embargo su valides respecto a restricciones del problema es correcta.

4.4. Arquitectura

Esta sección describe la arquitectura aplicada para poder desarrollar el algoritmo Estimador y sus componentes. Se describen librerías y recursos que ayudaron a un desarrollo más ágil. La Sección 4.4.1 describe aspectos

técnicos y la Sección 4.4.2 detalla la arquitectura aplicada para desarrollar Estimador.

4.4.1. Librerías y recursos

La estructura global del proyecto se separa en los módulos Core, UI, Commons y Benchmark. En el módulo Core se encuentra la implementación del framework y el Estimador, aquí se encuentran las interfaces, entidades y APIs para poder ejecutarlo. En el módulo Benchmark se encuentran los casos de pruebas, validación y rendimiento aplicado a CPLEX, Estimador y CPLEX+Estimador. El módulo Commons brinda componentes y funciones utilizadas por los demás módulos. Esto evita duplicar funcionalidades. El módulo UI se describe en la siguiente sección. En cuanto a detalles técnicos, los módulos Core, UI, Commons y Benchmark están implementado en Java 8 y definidos a través del gestor de dependencias Maven 3 [2], ya que agiliza la compilación, gestión de dependencias y desarrollo.

La interfaz web incluida en el módulo UI se desarrolló utilizando la librería gráfica Vaadin 8 [3] ya que permite en el código integrar elementos web y de servidor. La compilación de este módulo genera un WAR que puede ser instalado en varios contenedores Java como Tomcat[4], Jetty[5] y JBoss[6]. También fue utilizada la librería D3js[7] para poder graficar diagramas de Gantt. La generación de archivos para los casos de prueba, validación y rendimiento se llevaron a cabo por medio de la librería FreeMarker[8] que permite generar archivos en base a templates.

La programación concurrente se realiza por medio de la librería Threadly[9] dado que ofrece componentes de alto rendimiento para la ejecución de tareas concurrentes. Las librerías utilizadas forman parte del gran ecosistema ofrecido por la plataforma Java y en gran medida ayuda a un desarrollo ágil y seguro, motivo por el cual fue elegido.

4.4.2. Core

El Framework elaborado otorga una forma genérica de resolver problemas combinatorios. Para poder utilizarlo se requiere al menos una heurística constructiva para generar soluciones, opcionalmente una rutina que será invocada por cada solución generada y la definición de ciertas propiedades específicas al problema a resolver. Con distintas configuraciones se puede obtener distintos algoritmos. Por ejemplo si se cuenta con una heurística constructiva aleatoria y un proceso de mejora que realiza una búsqueda local el Estimador se comporta como el algoritmo GRASP. De igual forma se pueden obtener otros algoritmos conocidos y que siguen una estructura de la forma construir-mejorar.

El objetivo fue crear un algoritmo fácilmente extensible en la FC a través de la adición de heurísticas constructivas. En la Figura 4.1 se muestran sus principales módulos: Model Domain, Factory y Engine. Cada una de estas capas ofrecen un conjunto de funcionalidades y entidades. Muchas de ellas tienen funciones abstractas y cual su implementación es necesaria ya que su semántica depende de la naturaleza del problema a resolver.

El módulo Model Domain resuelve aspectos de representación y modelado de datos. En él se encuentra la entidad AbstractSolution que abstrae una representación de datos y le asocia un peso (fitness). El peso es una manera de cualificar una solución y sirve para comparar la calidad de las soluciones con otras. Su uso es importante en los otros módulos ya que independiente de la representación de datos aplicada en un problema, a través del peso asociado es posible evaluar y comparar soluciones.

El módulo Engine se encarga resolver una instancia del problema y de generar una solución. Se compone por las entidades AbstractImproverEngine, AbstractSolutionEngine y BaseExecuter. Utilizando como referencia el Algoritmo 3 se procede a explicar cada una de estas entidades. La entidad AbstractImproverEngine modela la fase de mejora y es un componente que se puede utilizar o no. La función “getImprovedSolution” que contiene

[2] Maven 3 - https://maven.apache.org/
[3] Vaadin 8 - https://vaadin.com
[4] Tomcat - http://tomcat.apache.org/
[5] Jetty - https://www.eclipse.org/jetty/
[6] JBoss - http://www.jboss.org/
[7] D3js - https://d3js.org/
[8] FreeMarker - https://freemarker.apache.org/
[9] Threadly - https://github.com/threadly/threadly

recibe una solución candidata y genera otra más. Aquí se puede implementar una rutina de mejora de soluciones o una búsqueda local.

AbstractSolutionEngine se encarga de modelar la fase constructiva, de aplicar (si corresponde) la fase de mejora y de generar una solución. Esto se puede ver en la linea 32 del Algoritmo 3. Para poder utilizar AbstractSolutionEngine es necesario proveer las funciones abstractas "getCandidateSolution" y "processCandidateSolution". El resultado dado por getCandidateSolution posteriormente es procesado por processCandidateSolution para devolver finalmente un AbstractSolution (ver Figura 4.1). La razón de separar en dos funciones es para que el resultado intermedio dado por "getCandidateSolution" pueda ser generado de distintas formas y una sola implementación de processCandidateSolution pueda ser utilizada. Además por ser abstracta aloja libertad en la forma de construir soluciones, por ejemplo, se pueden implementar un proceso que genere soluciones deterministas o aleatorias. Esta entidad además ofrece la solución sin aplicar FM, al aplicar FM y el tiempo de ejecución de cada caso.

Por último BaseExecuter representa la rutina principal, se encarga de ejecutar, obtener y comparar todas las soluciones manteniendo la mejor encontrada. Esta entidad revisa automáticamente la arquitectura de la PC para determinar la cantidad de hilos de ejecución que se pueden utilizar para generar soluciones en simultaneo. Este módulo también cuenta con las funciones necesarias para configurar y ejecutar el algoritmo. Por medio de la entidad EngineCfg se puede definir un motor para generar soluciones. EngineCfg contiene un SolutionEngineFactory, opcionalmente un ImproverFactory y la cantidad de soluciones que se quieren generar. Al permitir agregar varios EngineCfg, BaseExecuter se encarga de preparar y ejecutar todos los algoritmos, disponiendo luego el tiempo de ejecución y la mejor solución obtenida.

En el módulo Factory se encuentran las factories [10] SolutionEngineFactory e ImproverFactory. Dado que pueden existir varias implementaciones de AbstractSolutionEngine y AbstractImproverEngine, las factories ayudan a ocultar los detalles de implementación y disponer solo de las funciones necesarias. Estas entidades no son implementadas ya que son provistas por la entidad Supplier de Java 8.

Tal como se visualiza en la Figura 4.1 varias de las entidades están parametrizadas por los tipos genéricos CANDIDATE, SOLUTION_REPR y SOLUTION. SOLUTION es una abreviación de AbstractSolution<SOLUTION_REPR> para no sobrecargar el diagrama UML, mientras que CANDIDATE y SOLUTION_REPR son efectivamente representaciones del problema especifico a resolver. Ambos tipos pueden ser iguales, sin embargo la razón de que hayan dos es que CANDIDATE puede utilizarse como resultado previo a la solución final. Al momento de aplicar este framework es necesaria entonces definir los tipos de datos CANDIDATE y SOLUTION_REPR.

Se provee aquí un framework nombrado que define una estructura genérica, extensible, capaz de aprovechar computadoras modernas e inspirado en la estructura del algoritmo GRASP para resolver problemas combinatorios. Tal solución define un conjunto básico de entidades el cual requieren que se implementen ciertas funciones para poder aplicarlo, el cual de fondo determinan el problema especifico que se quiere enfrentar.

4.5. Implementación

La Sección 4.4.2 propone una estructura genérica para resolver problemas de optimización mediante la definición de ciertas funcionalidades. El objetivo en esta sección es describir como el framework se aplica y las entidades creadas para poder resolver efectivamente el problema de vulcanización en FunsaCoop.

Para comenzar el framework necesita dos representaciones de datos para definir los parámetros SOLUTION_REPR y CANDIDATE. Estás pueden ser iguales pero para el problema de este proyecto se aplicaron distintas representaciones. A continuación se expone la representación utilizada para SOLUTION_REPR y CANDIDATE. La Sección 4.2 define un ordenamiento de la producción como un conjunto de heaters el cual cada uno se compone por un conjunto de asignaciones. Para esto se definieron dos entidades: HeaterAssignment y MoldAssignment. MoldAssignment modela una asignación y tiene asociado un único heater durante un periodo de tiempo. Una instancia de este contiene entre uno y dos moldes, la cantidad a producir, el periodo de producción y el MoldAssignment que le precede. La entidad HeaterAssignment modela un heater y contiene la secuencia ordenada por periodo de MoldAssignments que debe producir. Cada HeaterAssignment tiene un identificador asociado. Un ordenamiento de la producción queda definido por un conjunto de HeaterAssignments y esta es

[10] Define una interfaz para crear un objeto, pero dejando en manos de las subclases la decisión de qué clase concreta instanciar. Permite que una clase delegue en sus subclases la creación de objetos. También conocido como virtual constructor.

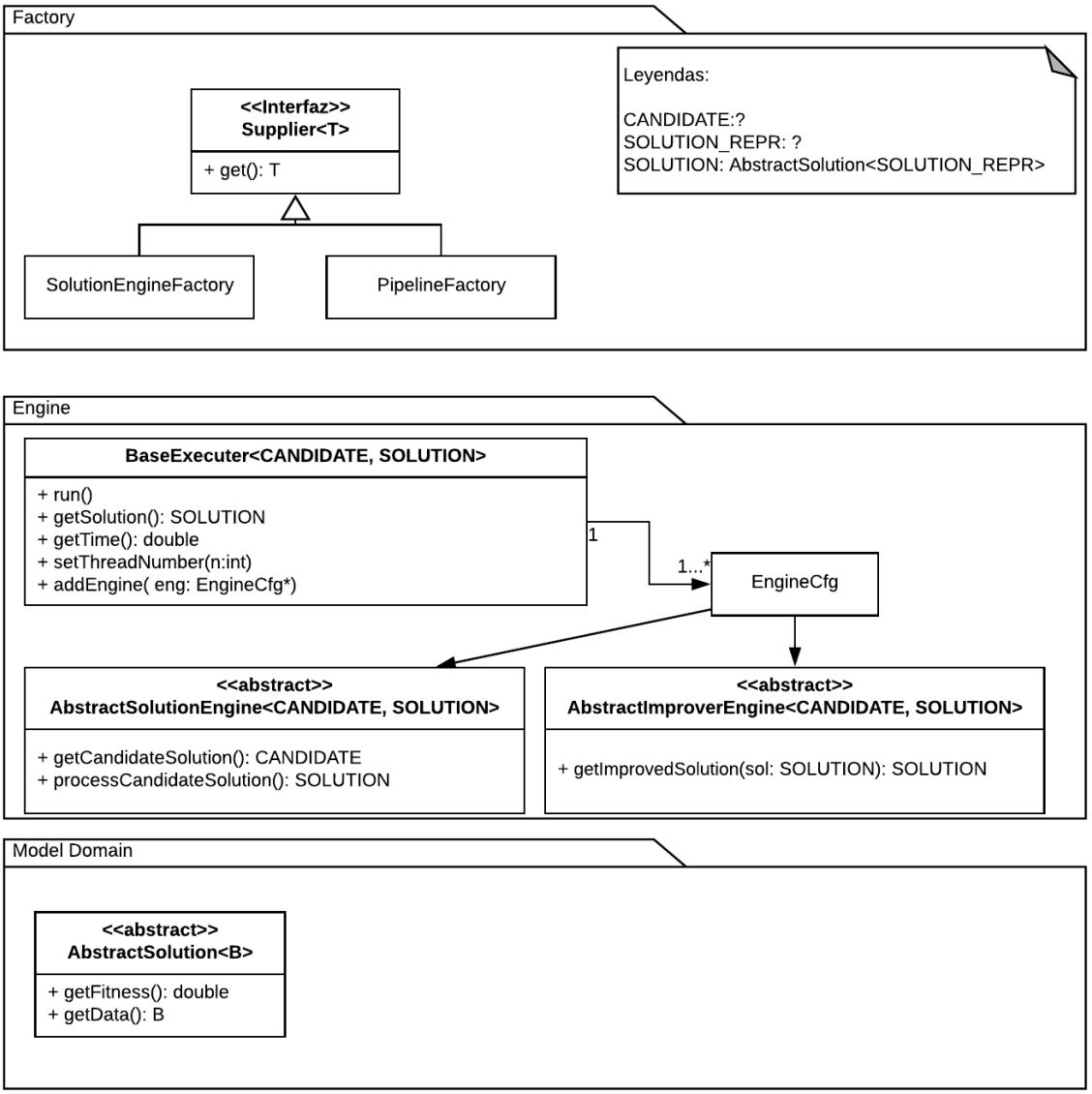


Figura 4.1: Arquitectura del Core

la representación utilizada para SOLUTION_REPR. Por otro lado se eligió representar CANDIDATE como un conjunto de MoldAssignments ya que esta representación es la utilizada al momento de particionar la demanda de cada molde.

El framework otorga la posibilidad de agregar varios procedimientos constructivos a través de AbstractSolutionEngine. Utilizando las representaciones definidas anteriormente se crea la entidad BaseHeuristic encargada de implementar las funciones "getCandidateSolution" y "processCandidateSolution" adicionando una función abstracta denominada "getNextAssignament". Esta función se aplica para generar la estructura definida para CANDIDATE y tiene como tarea seleccionar pares de moldes.

Utilizando BaseHeuristic se implementaron tres procedimientos heurísticos constructores de soluciones denominados Heurística Constructiva por Tiempo de Vulcanizado (HCTV), Heurística Constructiva Aleatoria (HCA) y Heurística Constructiva por Demanda (HCD). Como se explicó en la Sección 4.2 cada una de ellas define una forma de generar una solución según la instancia del problema variando la manera de seleccionar pares de moldes.

En la entidad SolutionImprover se encuentra la implementación de AbstractImproverEngine, el cual representa la fase de mejora. Por último se provee la entidad Executer basada en BaseExecuter, encargada de proveer funcionalidades extras como persistencia de soluciones y carga de instancias del problema desde archivos. Con las heurísticas constructivas y el proceso de mejora se obtiene el "Estimador" encargado de resolver el problema de FunsaCoop. Un punto importante es que es posible obtener distintas configuraciones para el Estimador según las heurísticas constructivas habilitadas y el proceso de mejora. Por ejemplo si solo se utiliza HCA y el proceso de mejora, cada ejecución del Estimador puede generar una solución distinta para una misma instancia del problema. En cambio si se utiliza HCTV cada ejecución genera la misma solución.

4.6. Interfaz gráfica web

Considerando que la tarea de crear una instancia del problema puede llegar a ser tediosa, reiterativa y la invocación del algoritmo Estimador también, se ofrece a través de una interfaz gráfica web un medio más amigable para poder llevarla a cabo. Este componente gráfico se encuentra en el módulo UI y tiene como finalidad facilitar la carga y edición del modelo de datos y visualización de resultados. Tal interfaz web incluye un conjunto de vistas que permiten configurar los parámetros de ejecución del algoritmo, cargar o guardar modelos de datos y visualizar los resultados de ejecución a través de la API dada por el modulo Core (Figuras 4.2, 4.3).

La vista inicial cuenta con dos paneles. El primer panel (Figura 4.2) permite cargar y guardar instancias del problema para ser resueltos. Esto evita al usuario el trabajo de ingresar todos los parámetros necesarios nuevamente. Además se incluye la posibilidad de habilitar/deshabilitar las heurísticas constructivas elaboradas y el proceso de mejora. También es posible configurar la cantidad de soluciones en simultaneo que el Estimador puede generar. Esto depende de la capacidad del hardware disponible.

El otro panel (Figura 4.3) se encarga de mostrar la planificación de la producción obtenida por el Estimador. Aquí está la posibilidad de guardar y abrir el resultado si necesidad de volver a ejecutar el algoritmo. La planificación se muestra como diagrama de Gantt. Cada segmento describe en una terna que moldes se deben utilizar y por cuanto periodos. El valor del periodo incluye los tiempos de setup y de producción.

Las otras vistas permiten definir una instancia del problema para resolver. A la izquierda de las Figuras (4.2, 4.3, 4.4) bajo el elemento "Editar instancia" se listan todos los campos requeridos por el modelo junto a su vista de edición. Entre ellos se encuentran:

Valores globales incluye nombre de instancia, duración de jornada laboral, cantidad de moldes, heaters y piezas.

Nombre de componentes el cual permite nombrar moldes, heaters y piezas para mejorar referencias en el diagrama de Gantt.

Disponibilidad de moldes configura la cantidad de moldes disponibles de un mismo tipo.

Disponibilidad de piezas es análogo a "Disponibilidad de moldes".

Requerimientos de piezas configura para cada molde las piezas que requiere.

Compatibilidad entre moldes configura que moldes se pueden procesar juntos en un heater.

Compatibilidad entre heaters y moldes configura que moldes se pueden procesar en un heater determinado.

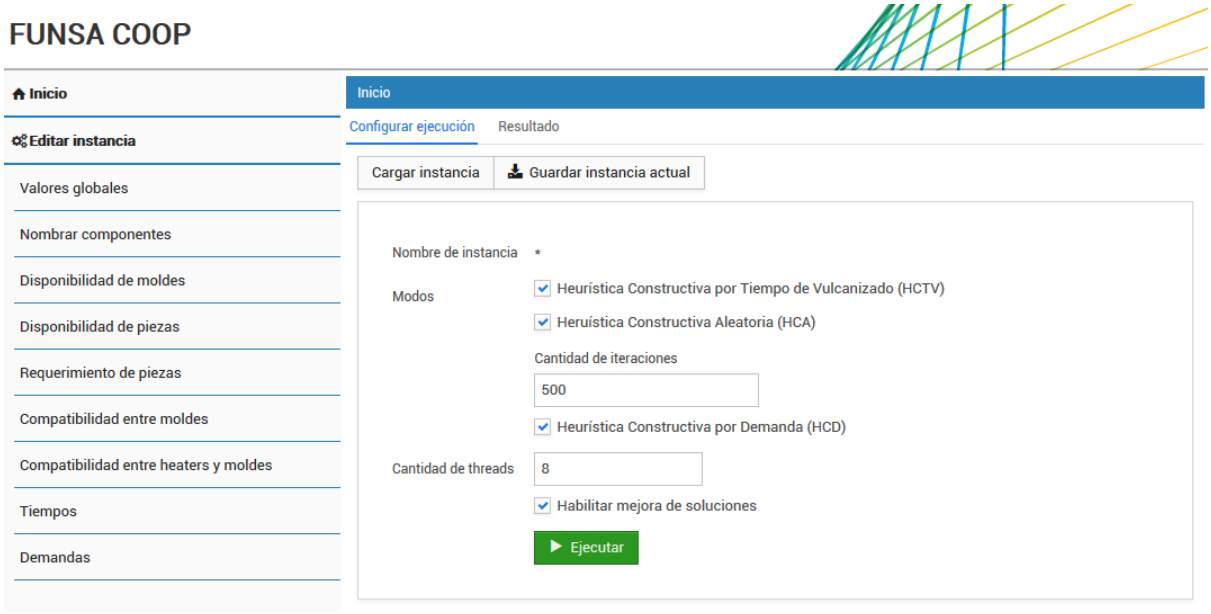


Figura 4.2: Solución informática, vista inicial.

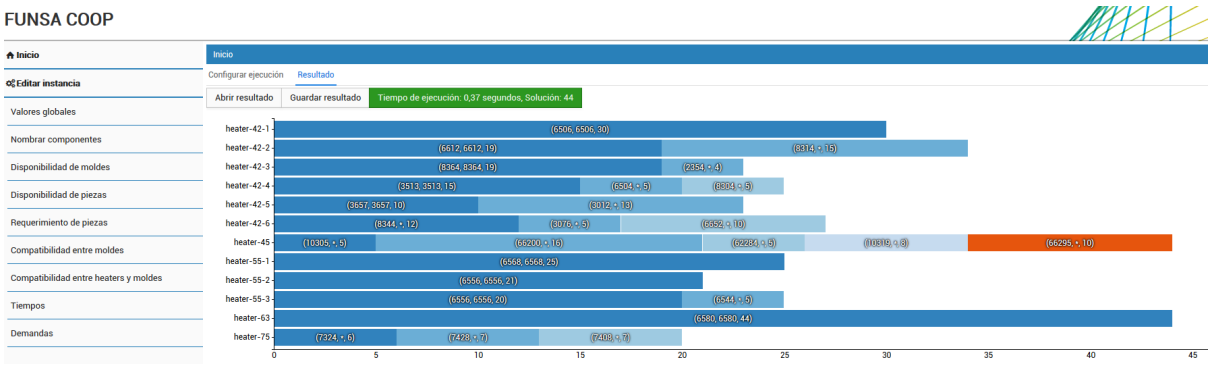


Figura 4.3: Solución informática, vista de resultados.

Tiempos configura los tiempos de colocado, vulcanizado y quitado de cada molde.

Demandas configura la demanda por cada tipo de molde.

Una vez que el usuario da "Ejecutar" (Figura 4.2) se lleva a cabo un proceso de validación de los datos ingresados. En caso de que los datos no sean válidos se indica al usuario. Por ejemplo en la Figura (4.5) el fallo dado por la etapa de validación indica que al menos es requerido que un molde sea compatible consigo mismo. Una vez que los datos son válidos se ejecuta el algoritmo mostrando un diagrama de Gantt, donde se especifica para cada heater los pares de molde a producir, el periodo de comienzo/fin y la cantidad a producir.

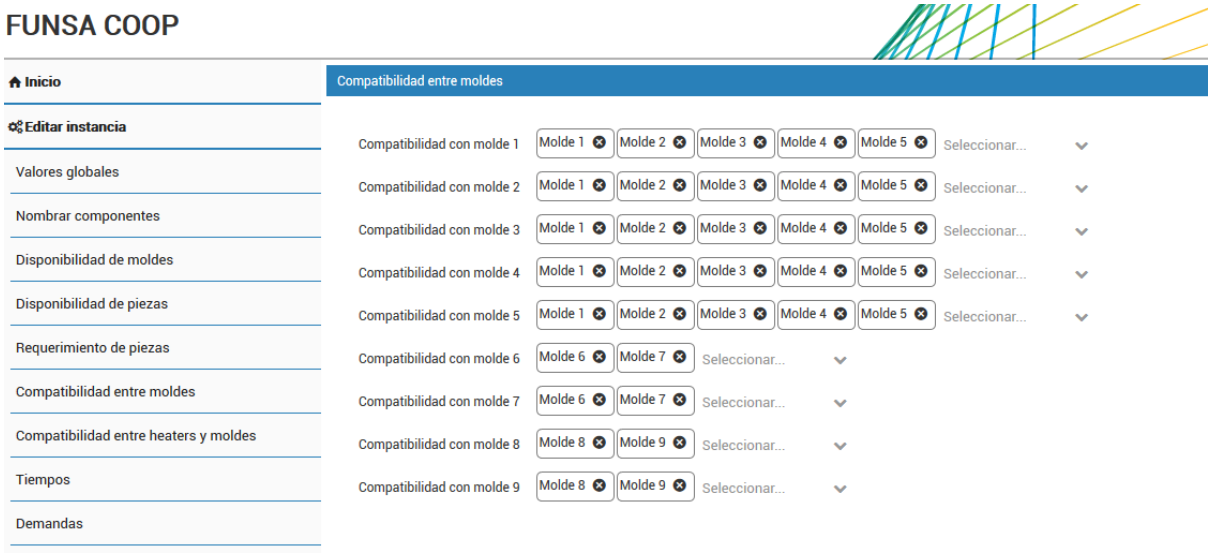


Figura 4.4: Solución informática, edición del modelo de datos.

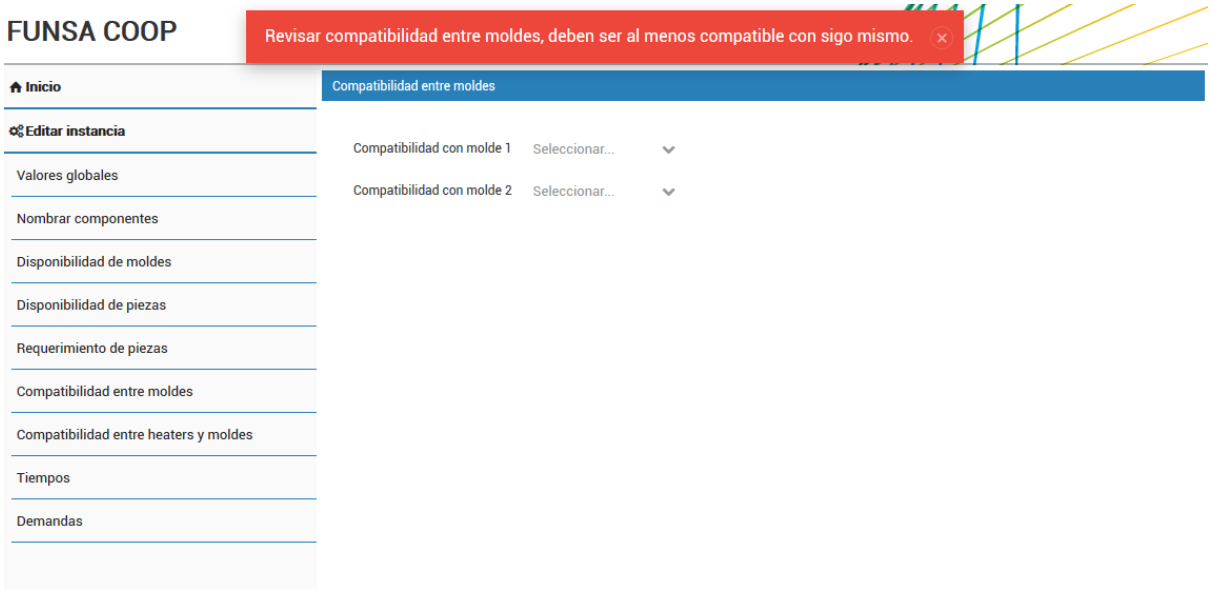


Figura 4.5: Solución informática, validación del modelo de datos.

CAPÍTULO 5

Experimentación numérica

5.1. Introducción

El Capítulo 4 expone tres procedimientos capaces de resolver el problema de ordenamiento en la producción de neumáticos. El primer procedimiento aplica técnicas exactas a través del software comercial CPLEX, el cual resuelve un modelo codificado en el lenguaje AMPL. Esta herramienta aplica una variedad de técnicas para reducir la cantidad de variables y restricciones que un modelo genera. Sin embargo la complejidad de este puede generar demoras importantes en cuanto a la resolución dado que este software busca soluciones óptimas.

El segundo procedimiento nombrado como Estimador aplica técnicas heurísticas para explorar el espacio de búsqueda. Entre sus ventajas está la velocidad de resolución por el uso de heurísticas para analizar el problema. No obstante las soluciones dadas no son siempre óptimas, sino que aproximadas. Se elaboraron tres heurísticas constructivas nombradas como HCTV, HCA y HCD que se pueden utilizar en el Estimador, además se puede complementar con un proceso de mejora.

El último procedimiento elaborado y nombrado como CPLEX+Estimador combina técnicas exactas y de aproximación. El objetivo es utilizar el Estimador para generar una estimación más adecuada del parámetro THB. De esta forma se puede reducir el tamaño del modelo que CPLEX debe resolver. En los resultados numéricos se ve una mejora importante respecto a los tiempos de ejecución dados por CPLEX.

Estas tres formas de resolución del problema reflejan que tan complejo puede llegar a ser, como un software comercial se comporta, que tan buena son las heurísticas constructivas aplicadas a Estimador y que provecho se le puede sacar si se combina con un software comercial. Para generar idea de que tan buenas son las distintas versiones desarrolladas en cuanto a calidad de solución y tiempos de ejecución este capítulo evalúa estos atributos mediante casos de pruebas y análisis de rendimiento. Adicionalmente se compara una planificación realizada por trabajadores de FunsaCoop con una generada por el Estimador.

En la Sección 5.2 se presenta el plan de pruebas que explica los pasos seguidos en la experimentación numérica. La Sección 5.3 explica las instancias del problema utilizadas y las columnas de las tablas de resultados. La Sección 5.4 se realiza el análisis de los resultados, comparando heurísticas, aplicación del proceso de mejora, CPLEX y CPLEX con Estimador, finalizando con conclusiones.

5.2. Plan de pruebas

El objetivo del plan de pruebas es definir los pasos a seguir para la obtención de los resultados. Para comenzar se definieron las instancias del problema a evaluar el cual consiste en once escenarios. Cada uno de ellos se compone por diez casos cada uno. En cada escenario se abarcaron datos reales, modificaciones de los mismos y de estrés. La variedad en los casos de prueba ayuda a realizar un análisis más objetivo y amplio de los procedimientos desarrollados. Se incluye además un escenario de estrés con diez casos para aplicar al Estimador. En la siguiente sección se detallan.

Al momento de contar con las instancias a evaluar, el siguiente paso consistió en encontrar la mejor configuración para el Estimador. Aquí se necesita configurar la cantidad de iteraciones, cantidad de hilos de ejecución, las heurísticas habilitadas y si se habilita o no el proceso de mejora. Se proponen ochos configuraciones basadas en que heurísticas se utiliza y si se aplica o no el proceso de mejora (FM):

1. HCTV
2. HCTV + FM
3. HCA
4. HCA + FM
5. HCD
6. HCD + FM
7. HCTV + HCA + HCD (HC4)
8. HCTV + HCA + HCD + FM (HC4 + FM)

Comenzando con la cantidad de iteraciones, las configuraciones que solo aplican heurísticas constructivas deterministas (1, 2, 5, 6) una iteración es suficiente dado que para una instancia del problema siempre generan la misma solución. En el resto de las configuraciones donde interviene la heurística constructiva aleatoria (3, 4, 7, 8) se considera suficiente 500 iteraciones (esto equivale a generar 500 soluciones). Este valor se obtuvo en pruebas previas realizadas.

En el uso de heurísticas que exploran aleatoriamente el espacio de soluciones es relevante calcular la media y desviación estándar para así conocer el promedio y dispersión de las soluciones generadas. Para esto se decidió correr 10 veces cada uno de los casos que apliquen la heurística aleatoria HCA. Respecto a la cantidad de hilos de ejecución, se configura el Estimador para que utilice 8 hilos ya que la PC utilizada tiene las características de la Tabla 3.2. Esto significa que el Estimador tiene la capacidad de generar 8 soluciones en simultaneo.

Contando con la mejor configuración para el Estimador, se procede a configurar el solver CPLEX. Inicialmente no estaba planeado cambiar ninguna configuración del solver, pero en la ejecución de varios casos se observó que algunas instancias del problema el solver CPLEX no es capaz de dar una solución exacta en un tiempo razonable. Muchas pasaban las 8 horas de ejecución sin obtener solución. Por lo tanto se decidió modificar algunos de sus parámetros para tratar de obtener alguna solución prematura. Uno de ellos es el "timeout", que representa el tiempo de procesamiento que tiene CPLEX disponible. En caso de exceder ese tiempo la ejecución termina y se devuelve la solución encontrada hasta el momento. Este valor se establece a 3600 segundos (una hora). Otro de los problemas presentado es la terminación de la ejecución por escasez de memoria debido a las dimensiones de las instancias generadas. Para tratar de resolverlo se utiliza el parámetro "nodefile" que determina el modo que CPLEX utiliza la memoria. El valor utilizado es 3, produciendo que las estructuras de datos utilizadas se compriman y se almacenen en disco cuando sea necesario. Este parámetro ayuda a que algunos casos de pruebas sea posible obtener una solución, pero no resuelve el problema de memoria completamente.

En resumen, el plan de pruebas elaborado es el siguiente. En un primer etapa se define los casos de pruebas y configuraciones de ejecución. Enseguida se ejecuta los casos de prueba para determinar que configuración del Estimador genera mejor soluciones. Luego se ejecutan los tres procedimientos de solución propuestos (CPLEX, Estimador y CPLEX+Estimador) utilizando los casos de pruebas definidos. Por último se ejecutan los casos de estrés definidos solo para el Estimador. Cada uno de los casos se corren empleando la PC con las especificaciones de la Tabla 3.2.

5.3. Descripción de instancias y estructura de resultados

Como se menciona en la Sección 5.2 referente al plan de pruebas se definieron once escenarios, cada uno de ellos compuesto por diez casos. Cada caso se genera a partir de un escenario base variando la demanda de los moldes en un rango $\pm 20\,\%$ respecto al valor original. Entre las instancias se incluyen casos reales obtenidos gracias al aporte de los trabajadores de FunsaCoop, también se incluyen casos de pruebas con modificaciones para así incrementar complejidad y estrés al resolverlos.

Los escenarios del uno al cinco se caracterizan por ser chicos, se aplican pocos moldes y la demanda de cada molde es baja, rondan en las 200 unidades. Entre las variaciones se incluyen el uso de moldes duplicados, piezas

compartidas, adición de más tipo de moldes y heaters. Los escenarios del seis al nueve simulan casos más reales, basados en los datos recolectados, también se incluyen algunas variaciones como las mencionadas anteriormente. Por último los escenarios diez y once están pensados como pruebas de estrés a través de un incremento importante en las demandas, adicionando piezas compartidas y más moldes de cada tipo. Estos incrementos de demandas, moldes y piezas compartidas dificultan el problema debido a que aumentan la cantidad de combinaciones a analizar. Las Tablas C.1 hasta la C.11 anexadas exponen los parámetros y valores utilizados en las pruebas realizadas para generar las instancias del problema.

Se incluye además un escenario con diez casos utilizados como prueba de estrés en Estimador. Los casos son generados con los parámetros de la Tabla C.12. Incluye 9 moldes el cual hay 20 de cada tipo y con una demanda que ronda en los 6 millones de unidades por cada molde. La cantidad de heaters disponibles se incrementa de a 5 unidades por caso, partiendo de 5 a 50. Estas instancias no pueden ser resueltas en los procedimientos que utilizan CPLEX como solver. El objetivo es ver el comportamiento de las heurísticas y del proceso de mejora implementadas para ver el tiempo de procesamiento que requieren.

Según lo planeado, luego de contar con las instancias del problema se procede a buscar la mejor configuración para Estimador. Las Tablas (C.1, C.2, C.3) presentan los resultados obtenidos. Bajo cada posible configuración se visualiza la solución, tiempo y además se incluye la media y desviación estándar en las configuraciones que aplican la heurística HCA. Las celdas resaltadas indican la mejor solución hallada, lo cual la mejor configuración para el Estimador corresponde a combinar todas las heurísticas más la FM, por lo tanto es la utilizada en las siguientes pruebas.

El siguiente paso es ejecutar los procedimientos de solución CPLEX, Estimador y CPLEX+Estimador. Las Tablas (C.4, C.5, C.6) presentan los resultados obtenidos. Bajo las columnas CPLEX, ESTIMADOR y CPLEX+ESTIMADOR se muestran los siguientes datos:

CPLEX:

Tiempo: Tiempo en segundos que el solver de CPLEX utilizó para resolver una instancia del problema.

Periodo disponible: Periodos disponibles para producir lo demandado.

Solución: Solución obtenida por CPLEX.

CPLEX_GAP: Error relativo de la solución.

Estado: Cada instancia ejecutada por CPLEX devuelve un estado de ejecución. Los posibles estados son:

ok: La ejecución fue exitosa.

timeout: La ejecución se detuvo porque el tiempo de espera expiró.

oom: Por limitaciones de recursos de cómputos, en especial de memoria, el solver no puede resolver la instancia.

ESTIMADOR:

Tiempo: Tiempo empleado por el Estimador para hallar una solución.

Solución: Solución dada por el Estimador. Las celdas resaltadas indican que la solución hallada coincide con la solución exacta dada por CPLEX.

CPLEX+ESTIMADOR: Se utilizan las mismas columnas que CPLEX, adicionando EST_GAP dando el error relativo entre la solución de CPLEX+Estimador y Estimador.

Por último la Tabla C.7 muestra el tiempo de ejecución y solución de los casos pertenecientes al escenario de estrés.

5.4. Análisis de resultados

5.4.1. Introducción

Esta sección se destina a analizar los resultados obtenidos. La Sección 5.4.2 compara los resultados de las heurísticas constructivas HCTV, HCA y HCD sin aplicar proceso de mejora. Esto permite ver en función de los

escenarios el comportamiento de cada una de ellas. La Sección 5.4.3 compara cada heurística con el proceso de mejora habilitado. Esta comparación refleja que tan bueno fue aplicar el proceso de mejora respecto a calidad de solución y el tiempo de resolución. En la Sección 5.4.4 se combinan todas los heurísticas constructivas sin y con proceso de mejora. El objetivo es revisar el rendimiento del Estimador cuando cuenta con un conjunto de heurísticas constructivas para utilizar. Por último en la Sección 5.4.5 se compara CPLEX, Estimador y CPLEX+Estimador.

5.4.2. Comparación de heurísticas constructivas sin proceso de mejora

Se propusieron 3 heurísticas constructivas aplicadas en la fase constructiva del Estimador. Cada una de ellas define una política de agrupación de moldes, obteniendo así distintos comportamientos según la instancia del problema. En la Figura 5.1 se compara el promedio de soluciones dadas por las heurísticas HCTV, HCA y HCD en cada escenario.

Se puede ver que HCTV superó en calidad de solución en casi todos los escenarios, luego le sigue HCA y HCD. Esto es porqué HCTV agrupa moldes compatibles y con menor diferencia en tiempo de vulcanizado, esto permite producir más moldes en menos tiempos. Los escenarios que cuentan con varios moldes del mismo tipo HCTV tiende a generar buenas soluciones ya que se generan asignaciones con mismos tipos de moldes e igual tiempo de vulcanizado, logrando así producir más neumáticos. Por otro lado HCA solo selecciona pares de moldes que sean compatibles, no analiza ninguna otra característica del problema. No obstante su potencial está en la capacidad de generar distintas soluciones. Esto permite ampliar la exploración del espacio de búsqueda y contar con una variedad mayor de soluciones candidatas. Por último HCD fue el que tuvo en promedio peores soluciones. Sin embargo se puede ver una característica que analiza esta heurística: en los escenarios donde el promedio se asemeja al de HCTV se debe a que hay más de un molde del mismo tipo disponible. HCD los agrupa ya que la diferencia de demanda es cero y esto permite producir más en menos tiempo. En los demás escenarios se cuenta por lo general con un molde de cada tipo y la diferencia entre las demandas produce soluciones peores a las otras heurísticas.

En cuanto a los tiempos de ejecución, ninguna heurística superó los dos segundos para generar una solución. Mucho se debe a como están implementadas las heurísticas constructivas. Por un lado HCTV solo debe encontrar la menor diferencia en tiempo de vulcanizado entre moldes compatibles, y en caso de que sea cero ya detiene la búsqueda. En el peor de los casos la complejidad es cuadrática. HCA solo debe seleccionar un par de moldes aleatoriamente y no mucho más, ya que la mayoría de la información del problema está dada. HCD aplica la misma metodología que HCTV salgo que busca la menor diferencia por demanda, y la complejidad de la operación es cuadrática.

Esto demuestra que la simplicidad y las estructuras de datos usadas internamente permitieron lograr consultas y construcción de soluciones de forma más rápida. El uso de generación simultanea de soluciones como forma de aprovechar el hardware moderno también ayudó como una forma de explorar más rápido el espacio de búsqueda. Ninguna de las heurísticas propuestas (HCTV, HCA, HCD) es mejor que otra, cada una analiza características específicas de la instancia del problema, por lo que su solución puede ser muy buena en algunos escenarios o muy malas en otros. Dado que la complejidad de cómputos no es grande, se aconseja utilizarlas a todas como forma de ampliar la búsqueda de soluciones. En la Tabla 5.2 se pueden ver los promedios de soluciones y tiempos de ejecución para cada escenario.

5.4.3. Comparación de heurísticas constructivas con proceso de mejora

Además de las heurísticas constructivas se elaboró un proceso de mejora que complementa al Estimador. Tal proceso busca corregir y ajustar las soluciones generadas por las heurísticas constructivas elaboradas. Las Figuras 5.3, 5.4, 5.5 muestra la mejora obtenida cuando se aplica el proceso de mejora en cada heurística constructiva.

HCA mostró el mayor porcentaje de mejora. Gracias al proceso de mejora las soluciones en HCA mejoraron un 17 %. Esto es porque HCA explora el espacio de búsqueda en forma aleatoria y la estructura de las soluciones dan lugar a más correcciones y ajustes para reducir el tiempo total. Agrupar moldes compatibles sin un análisis exhaustivo genera una gran variedad de soluciones el cual muchas de ellas pueden ser mejoradas. Por otro lado en HCD se pudo mejorar hasta un 15 % las soluciones y en HCTV un 4 %. Estos valores no indican que tan mala es una heurística constructiva ya que el proceso de mejora interviene en algunas zonas de la solución. Específicamente se analizan algunas asignaciones intermedias y las ubicadas en la cima de cada pila. Por esto la

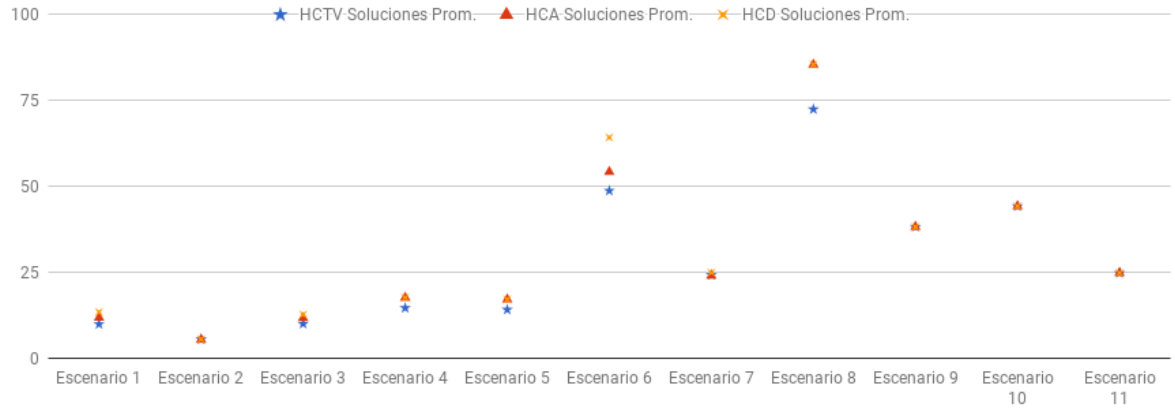


Figura 5.1: Soluciones promedio en heurísticas constructivas sin proceso de mejora

	HCTV		HCA		HCD	
	Soluciones Prom.	**Tiempo Prom.**	**Soluciones Prom.**	**Tiempo Prom.**	**Soluciones Prom.**	**Tiempo Prom.**
Escenario 1	10	0,00144	11,9	0,02566	13,5	0,0002
Escenario 2	5,6	0,00024	5,5	0,03241	5,6	7,00E-05
Escenario 3	10,1	0,00085	11,8	0,0149	12,8	5,00E-05
Escenario 4	14,7	0,00013	17,7	0,02526	17,7	8,00E-05
Escenario 5	14,2	0,00012	17,2	0,03785	17,2	0,00011
Escenario 6	48,8	9,00E-05	54,3	0,01381	64,3	4,00E-05
Escenario 7	24,3	0,00014	24,1	0,05228	25	0,00012
Escenario 8	72,5	0,0001	85,5	0,0446	85,5	8,00E-05
Escenario 9	38,3	0,00016	38,3	0,14953	38,3	0,00019
Escenario 10	44,3	6,00E-05	44,3	0,02782	44,3	7,00E-05
Escenario 11	24,9	0,00023	24,9	0,5832	24,9	0,0002

Figura 5.2: Soluciones promedio en heurísticas constructivas sin proceso de mejora

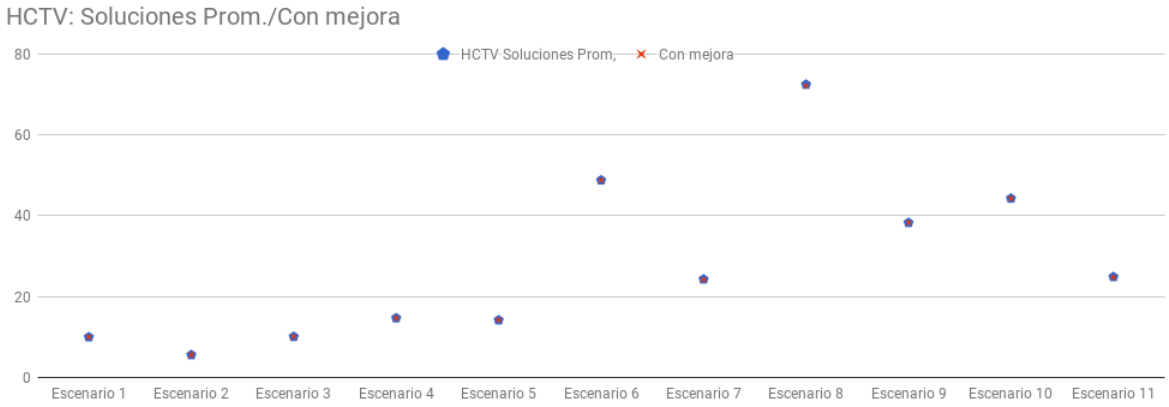


Figura 5.3: HCTV con proceso de mejora

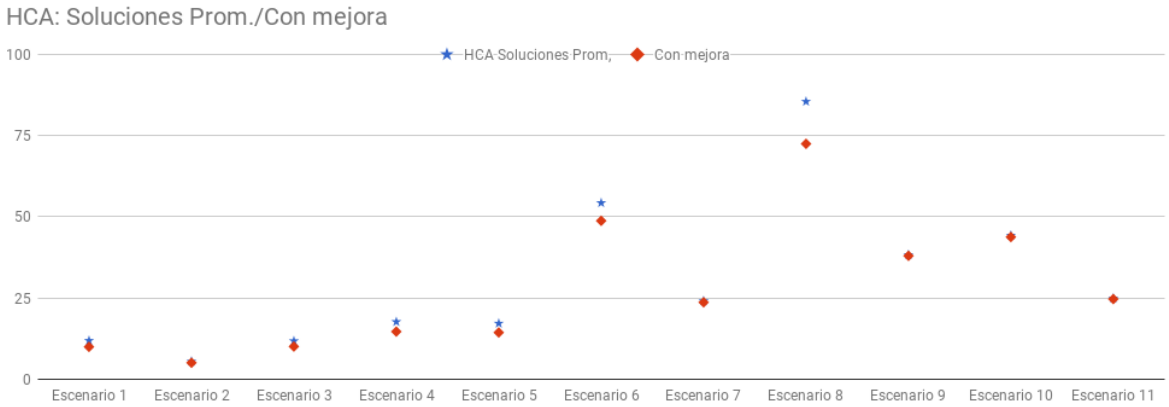


Figura 5.4: HCA con proceso de mejora

mejora depende en parte a la estructura de la solución. Esta fase se puede ver como una "distorsión" que se le aplica a la solución, en algunos casos puede generar mejoras y en otros puede hasta generar una solución peor.

Respecto al tiempo empleado, incluir este proceso no genera un incremento importante en la resolución. El mayor aumento fue de 10 veces respecto al tiempo original, pero se está hablando de 0.0002 a 0.00223 segundos. No se superó en ningún caso los dos segundos de ejecución para obtener una solución. En base a los resultados es recomendable incluir este proceso como complemento a las heurísticas constructivas elaboradas y otras que se quieran agregar. En las Tablas 5.6 y 5.7 se pueden ver el resumen de del promedio de soluciones y tiempos respectivamente.

5.4.4. Combinación de heurísticas constructivas

La configuración HC4 compuesta por HCTV, HCA y HCD aplicada al Estimador sirvió como una manera de evaluar la capacidad de exploración del espacio de búsqueda y procesamiento. En la medida que se agregan más componentes de búsqueda el algoritmo incrementa su capacidad de generar diversos tipo de soluciones. Pero según que tan exhaustivo sea el análisis aplicado puede influir negativamente en el tiempo de resolución.

No obstante la adición de heurísticas constructivas es una manera de ampliar el espacio de búsqueda y permite encontrar soluciones más próximas a las óptimas. Es bueno seguir la definición de heurística como rutinas sencillas y conceptuales. La PC utilizada cuenta con 8 cores e internamente el Estimador es capaz de detectar esta capacidad a través del framework elaborado. Esto permite generar soluciones de forma simultanea

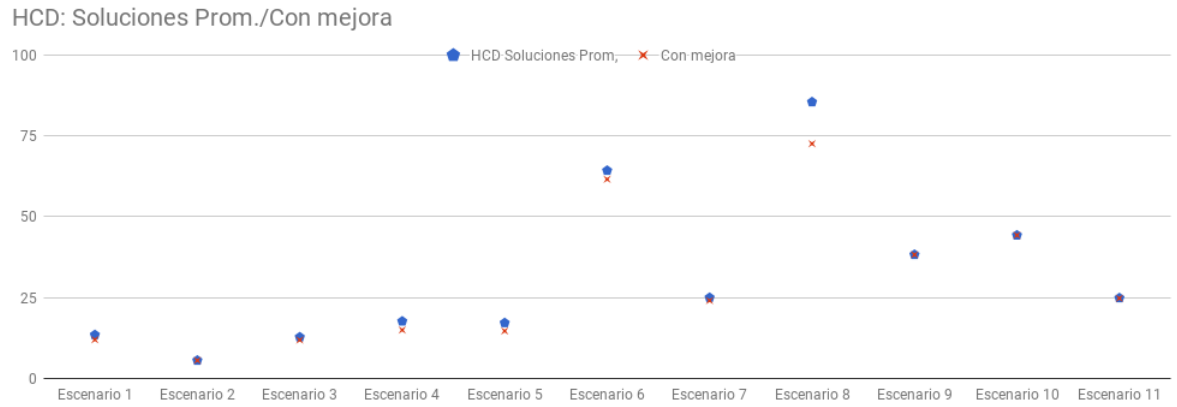


Figura 5.5: HCD con proceso de mejora

	HCTV			HCA			HCD		
	Soluciones Prom.	Con mejora	% de mejora	Soluciones Prom.	Con mejora	% de mejora	Soluciones Prom.	Con mejora	% de mejora
Escenario 1	10	10	0	11,9	10	0,1597	13,5	12,00	0,1111
Escenario 2	5,6	5,6	0	5,5	5,1	0,0727	5,6	5,60	0
Escenario 3	10,1	10,1	0	11,8	10,1	0,1441	12,8	11,90	0,0703
Escenario 4	14,7	14,70	0	17,7	14,7	0,1695	17,7	15,00	0,1525
Escenario 5	14,2	14,2	0	17,2	14,4	0,1628	17,2	14,70	0,1453
Escenario 6	48,8	48,80	0	54,3	48,8	0,1013	64,3	61,60	0,042
Escenario 7	24,3	24,2	0,0041	24,1	23,7	0,0166	25,00	24,10	0,036
Escenario 8	72,5	72,2	0,0041	85,5	72,5	0,152	85,5	72,60	0,1509
Escenario 9	38,3	38,3	0	38,3	38	0,0078	38,3	38,30	0
Escenario 10	44,3	44,30	0	44,3	43,8	0,0113	44,3	44,30	0
Escenario 11	24,9	24,8	0,004	24,9	24,7	0,008	24,9	24,80	0,004

Figura 5.6: Resumen de soluciones promedio de heurísticas constructivas

	HCTV			HCA			HCD		
	Tiempo Prom, (seg)	Con mejora	% de demora	Tiempo Prom, (seg)	Con mejora	% de demora	Tiempo Prom, (seg)	Con mejora	% de mejora
Escenario 1	0,00144	0,00215	0,4931	0,02566	0,02576	0,0039	0,0002	0,00029	0,45
Escenario 2	0,00024	0,00079	2,2917	0,03241	0,03255	0,0043	7e-05	0,00013	0,8571
Escenario 3	0,00085	0,00088	0,0353	0,0149	0,01497	0,0047	5e-05	0,00016	2,2
Escenario 4	0,00013	0,00016	0,2308	0,02526	0,02542	0,0063	8e-05	0,00031	2,875
Escenario 5	0,00012	0,00013	0,0833	0,03785	0,03809	0,0063	0,00011	0,00042	2,8182
Escenario 6	9e-05	0,0001	0,1111	0,01381	0,0139	0,0065	4e-05	0,00014	2,5
Escenario 7	0,00014	0,00024	0,7143	0,05228	0,05262	0,0065	0,00012	0,00019	0,5833
Escenario 8	0,0001	0,00011	0,1	0,0446	0,04491	0,007	8e-05	0,00038	3,75
Escenario 9	0,00016	0,00034	1,125	0,14953	0,15053	0,0067	0,00019	0,00044	1,3158
Escenario 10	6e-05	0,00017	1,8333	0,02782	0,02799	0,0061	7e-05	0,00015	1,1429
Escenario 11	0,00023	0,00261	10,3478	0,6347	0,63921	0,0071	0,0002	0,00223	10,15

Figura 5.7: Resumen de tiempos promedio de heurísticas constructivas

HC4		
Soluciones Prom.	HC4 con mejora	% de mejora
10	10,00	0
5,5	5,10	0,0727
10,1	10,10	0
14,7	14,70	0
14,2	14,20	0
48,8	48,80	0
24,10	23,70	0,0166
72,5	72,50	0
38,3	38,00	0,0078
44,3	43,70	0,0135
24,5	24,50	0

Figura 5.8: Comparación de heurísticas con y sin proceso de mejora

y en menos tiempo. Aplicar las 3 heurísticas constructivas en todos los escenarios sin fase de mejora no superaron los dos segundos de ejecución.

La configuración "HC4 + FM" demostró comportarse de forma positiva al no generar incremento en los tiempos de ejecución, el cual no superó tampoco los 2 segundos en todos los escenarios. Además aplicar esta fase ayudó a mejorar las soluciones en promedio un 7 %. Se esperaba que esta configuración fuera la mejor dado que todas las heurísticas elaboradas en conjunto se complementan y explorar más soluciones para el problema de FunsaCoop. La aplicación de rutinas sencillas combinadas con el uso de estructuras de datos eficientes y la capacidad de hardware permitió elaborar heurísticas eficiente. La Tabla 5.8 muestra el promedio de soluciones y tiempo cuando se combinan todas las heurísticas constructivas.

5.4.5. Análisis de CPLEX y Estimador

Parte de los resultados permite ver hasta que punto las técnicas exactas pueden aplicarse y en que punto las de heurísticas muestran su potencial. El Capítulo 4 describe dos procedimientos de solución que utilizan el solver CPLEX, una de ellas lo aplica directamente y la otra se utiliza junto con el Estimador. También se describieron los problemas de utilizar este solver para resolver la formulación matemática desarrollada. Uno de ellos está vinculado a la diferencia entre THB y la solución óptima de la instancia del problema. Si esta diferencia es grande (en el orden del 25 % de la solución óptima) CPLEX debe explorar un espacio de soluciones aún mayor. Además si se dificulta el caso de prueba incrementando la cantidad de moldes o la demanda se obtienen estados como timeout o out-of-memory por parte de CPLEX. En este tipo de instancias el solver no es capaz de generar una solución óptima, sino que una prematura o ninguna.

En la Tabla 5.9 se compara CPLEX y Estimador con la configuración H4+FM. Las celdas con asterisco referente a la Figura 5.9 indica que CPLEX no fue capaz de obtener una solución. Uno de los principales motivos es la estimación del parámetro THB el cual es un valor muy alejado del óptimo. En escenarios donde se cuenta con más cantidad de moldes y demandas mayores el estimador dado por la ecuación 4.1 genera valores muy grandes cuando no corresponde. Otro punto en contra es que los escenarios marcados con asterisco son los obtenidos del relevamiento de datos hechos en FunsaCoop, por lo que este procedimiento no podría aplicarse en escenarios reales.

Entre las diferencias importantes que se pueden ver son los tiempos de ejecución. CPLEX busca soluciones óptimas a través de un análisis más exhaustivo el cual conlleva más tiempo. Por otro lado Estimador demostró que es capaz de generar soluciones óptimas en algunos escenarios y en menos de un segundo. Esto refleja una diferencia común entre las técnicas exactas y de aproximación. Se puede observar que algunos valores de la columna "Solución" (Tabla 5.9) del Estimador son menores al de CPLEX. Esto es porque en algunos casos CPLEX detuvo su ejecución y devolvió una solución prematura (no óptima) y Estimador encontró una mejor. Esto no indica que la solución de Estimador es óptima.

En la Tabla 5.9 también se compara CPLEX+Estimador (correspondiente al tercer procedimiento presentado en el Capítulo 4) y Estimador con la configuración H4+FM. La combinación CPLEX+Estimador se propuso para resolver en parte los problema de utilizar solo CPLEX. A través del Estimador se pueden obtener valores para THB cercanos a la solución óptima del problema y esto genera un modelo más ajustado para que el solver

CPLEX resuelva. Esta combinación redujo el exceso de variables innecesarias del modelo. Con un modelo más chico se mejoró notoriamente los tiempos de CPLEX y permitió obtener más soluciones óptimas dado que el parámetro THB tiene un valor más razonable y cercano a la solución óptima. De los 110 casos ejecutados se obtuvieron 49 soluciones óptimas al utilizar solo CPLEX, 100 se obtuvieron al combinarlo con Estimador, se obtuvo una mejora del 104 %. Este incremento de soluciones óptimas permitió comparar mejor las dadas por el Estimador con la configuración HC4+FM. En las Tablas C.4,C.5, C.6 se puede ver que de los 110 casos ejecutados, 89 corresponden a soluciones óptimas y en las que no el error relativo es bajo, mostrando que a pesar de no ser una solución óptima, esta es aproximada.

Cabe resaltar que inicialmente se utilizaron instancias del problema mucho más grandes el cual CPLEX no era capaz de resolverlas, lo cual no era posible comprar los distintos procedimientos. Ante esta situación se decidió crear instancias más chicas pero manteniendo casos reales y algunos de estrés. Por eso se crea un escenario de estrés para probar el Estimador con instancias grandes del problema. Los resultados de la Tabla C.7 muestran que para instancias de 6 millones de moldes demandados en promedio, con 50 heaters y 20 unidades por cada tipo de molde el tiempo requerido no superó los 20 segundos. Esto demuestra que el desarrollo de heurísticas con una lógica específica a distintas clases de instancias sirve para obtener buenos tiempos de ejecución.

Respecto a los tiempos de ejecución de CPLEX se puede observar que algunos escenarios sencillos (como el 4, 5 y 6) demoran en promedio más que escenarios más complejos (como el 9, 10 y 11), cuando debería intuitivamente ser al revés. Una posible causa es por la forma que CPLEX trata de encontrar una solución. Los casos más complejos disponen de más moldes del mismo tipo y existen patrones de soluciones que si se aplican reiteradamente pueden llevar a soluciones óptimas. Un ejemplo es producir un par de molde del mismo tipo el mayor tiempo posible. Por lo tanto es más sencillo generar varios de estos patrones "simples", ubicarlos en el ordenamiento y luego analizar un conjunto mucho más reducido de combinaciones. En los casos sencillos no se pueden aplicar patrones que reduzcan las combinaciones a analizar, por lo tanto el solver CPLEX requiere hacer un análisis más exhaustivo e implica más tiempo de cómputos. Esto podría argumentar algunos de los casos donde se producir timeout o out-of-memory en las ejecuciones de CPLEX.

5.5. Evaluación de escenario real

La planificación de la producción en FunsaCoop es un trabajo que se ha realizado de forma manual y en base a planificaciones similares. Además esta no es estática ya que pueden surgir pedidos de clientes que cambien las prioridades de entrega de neumáticos. Por lo tanto pueden surgir cambios de planificación en la medida que se ejecuta. Gracias al departamento de Planificación y Control de la Producción (P.C.P.) de FunsaCoop fue posible obtener la planificación de vulcanizado realizada para un pedido hecho por clientes de Venezuela. Toda la información relevante para planificar se encuentra en una planilla Excel compuesta por 21 hojas de cálculo. Aquí se incluye calculo de horas, requerimientos de materia prima y piezas, cantidades demandadas, entre otros. El objetivo de esta sección es comparar la planificación realizada en FunsaCoop con la generada por Estimador y CPLEX.

La Figura 5.10 muestra la planificación de FunsaCoop. Cada una de las celdas coloreadas representa un turno de trabajo de 8 horas. En cada una se produce un par de moldes en un heater determinado. En FunsaCoop los heaters y moldes se configuran y dejan pronto para operar un día anterior. Además su planificación incluye varios factores que se debieron considerar a medida que la producción de neumáticos avanzaba. En este trabajo muchos de ellos tuvieron que ser asumidos ya que se trataría con un problema más complejo de resolver. Uno de ellos es la cantidad de trabajadores que se cuenta por turno, otro es la ubicación de piezas en la fábrica. También se debe considerar el stock de green tires e imprevistos que pueden surgir. Estos factores explican en parte las celdas no coloreadas entre asignaciones en la planificación de FunsaCoop (Figura 5.10).

Para comparar la planificación real con una generada por el Estimador se realizaron ajustes para llevarla a la realidad simplificada que aquí se asumió. La Figura 5.11 muestra la planificación ajustada de FunsaCoop. Cada segmento incluye el tiempo de configuración y de producción (por eso no se ven espacio en blanco entre segmentes, como en la Figura 5.10). En la planificación original se requieren aproximadamente 50 turnos, mientras que en la ajustada se necesitan 42.

Para poder utilizar CPLEX y el Estimador fue necesario definir la instancia del problema. En la planillas Excel se tiene registro de las demandas por cada tipo de neumático. Sabiendo que cada turno de trabajo dura 8 horas (480 minutos) y la cantidad de celdas utilizadas para producir un determinado neumáticos fue posible deducir los tiempos de vulcanizados de cada molde. El tiempo de colocado, quitado y limpieza fueron dados por

el departamento de P.C.P.. FunsaCoop además cuenta con un heater que solo puede albergar un molde. Para poder simular esto en la instancia elaborada los moldes compatibles con este heater no se consideran compatibles entre sí. De esta manera se asegura que solo un molde puede ser albergado. Los demás parámetros del problema como la compatibilidad entre moldes, moldes-heaters y la disponibilidad de moldes se obtuvieron principalmente de la planilla correspondiente a la Figura 5.10. La instancia definida requiere de 12 heaters y 25 tipos de moldes (35 contando con los duplicados), en la Tabla C.13 se muestran todos los valores utilizados.

Comenzando con CPLEX, se obtuvo que la solución óptima requiere de 42 turnos de trabajo. El tiempo requerido fue de 14 segundos aproximadamente. Se pudo ver que en la solución hallada hay una gran cantidad de asignaciones con periodos de producción chicos. A pesar que esta manera lleva a la solución óptima, en términos prácticos no se prefiere ya que se es mejor producir asignaciones en lotes grandes. Esto es una práctica de fabricación.

La ejecución con el Estimador se hizo con la configuración $H4 + FM$. Como resultado se obtuvo una planificación (Figura 5.12) que requiere 42 turnos de trabajo para satisfacer la demanda. Esta solución coincide con la de CPLEX, por lo tanto es exacta. El tiempo empleado para generar la planificación fue de 0,25 segundos aproximadamente. Esta solución no es muy buena en términos prácticos ya que en los heaters "heater-42" y "heater-45" hay muchas asignaciones que solo producen un molde. En FunsaCoop puede darse el caso de que un heater con dos compartimientos solo tenga para producir un molde, pero se trata de evitar lo más posible ya que el deterioro en ellos es mayor y además se desperdicia vapor de la caldera. Alternativamente con el Estimador se pudo generar otra planificación donde también requiere 42 jornadas para satisfacer la demanda. El tiempo empleado fue de 0,23 segundos aproximadamente. En esta planificación se puede ver que hay menos heaters con compartimientos vacíos y mejor aprovechamiento de la capacidad disponible.

El punto positivo de utilizar CPLEX es que encuentra una solución óptima para el problema, sin embargo su estructura en términos prácticos puede no ser adecuada. Además el comportamiento de CPLEX en cuanto a tiempo de ejecución puede variar según la complejidad de la instancia del problema. Por otro lado en Estimador se consideran más aspectos prácticos relevados de FunsaCoop, por lo que su soluciones se asemejan más a lo que se esperaría. Nos obstante las soluciones no siempre son óptimas. Un punto positivo es que en situaciones de estrés Estimador logra encontrar una solución en un tiempo más razonable.

5.6. Resumen

Si se cuenta con recursos de cómputos y tiempo se puede decir que la solución CPLEX+Estimador es una buena herramienta para resolver instancias de complejidad baja y media. Sin embargo se debe contar con un licencia comercial para poder utilizar CPLEX y AMPL como herramientas adicionales. Emplear solo el Estimador como herramienta puede servir de guía para obtener una planificación de producción ya que requiere menos recursos de cómputo, tiempo y es posible abarcar instancias más complejas. También tiene la capacidad de generar soluciones óptimas en instancias chicas y medianas, el cual se incluyen escenarios reales. En medida es bueno aplicar todas las heurísticas implementadas ya que se complementan entre sí y además incluir al proceso de mejora. Desde el punto de vista práctico, los ordenamientos generados por CPLEX tienen asignaciones muy "dispersas" entre los heaters, en el contexto de FunsaCoop esto puede implicar más traslado de piezas y maquinaria en toda la instalación. En cambio el algoritmo desarrollado está pensado para generar asignaciones homogéneas en todo el ordenamiento. No se pretende como parte del análisis juzgar el software CPLEX ya que es una herramienta capaz de resolver modelos matemáticos y el cual a veces es complicado representar realidades específicas del problema. CPLEX permitió generar un punto de validación y comparación utilizados en la formulación matemática y en la elaboración del framework junto al Estimador.

	CPLEX		Estimador (HC4+FM)			CPLEX+Estimador (HC4+FM)	
	Tiempo (seg)	Solución Prom.	Tiempo (seg)	Solución Prom.	CSE	Tiempo (seg)	Solución Prom.
Escenario 1	33,919	10	0,03349	10	10	1,62326	10
Escenario 2	13,755	5,1	0,03463	5,1	10	1,00773	5,1
Escenario 3	180,521	10,1	0,02178	10,1	10	1,56447	10,1
Escenario 4	258,4807	14,7	0,02728	14,7	10	1,86355	14,7
Escenario 5	499,9284	14,3	0,04208	14,2	10	2,37408	14,2
Escenario 6	1326,1297	72,8	0,01329	48,8	7	1142,36765	*
Escenario 7	721,6769	*	0,0608	23,7	10	3,74712	23,7
Escenario 8	55,4624	*	0,04761	72,5	9	483,60137	*
Escenario 9	28,0574	*	0,14751	38	7	418,62036	37,8
Escenario 10	90,6468	*	0,028	43,9	6	902,49752	*
Escenario 11	298,3171	648	0,58922	24,6	0	711,77416	21,2

Figura 5.9: Comparación de procedimientos propuestos (CSE: Cantidad de soluciones exactas.)

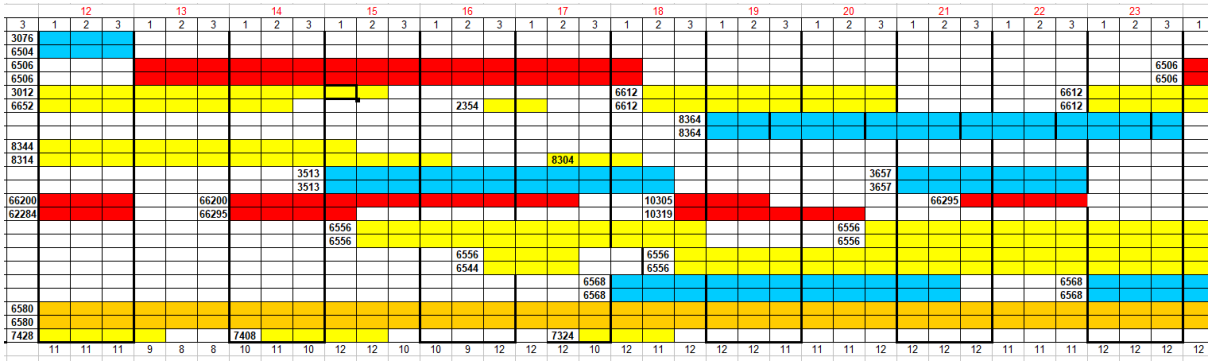


Figura 5.10: Planificación de vulcanizado de FunsaCoop (Fuente: Departamento de P.C.P. de FunsaCoop)

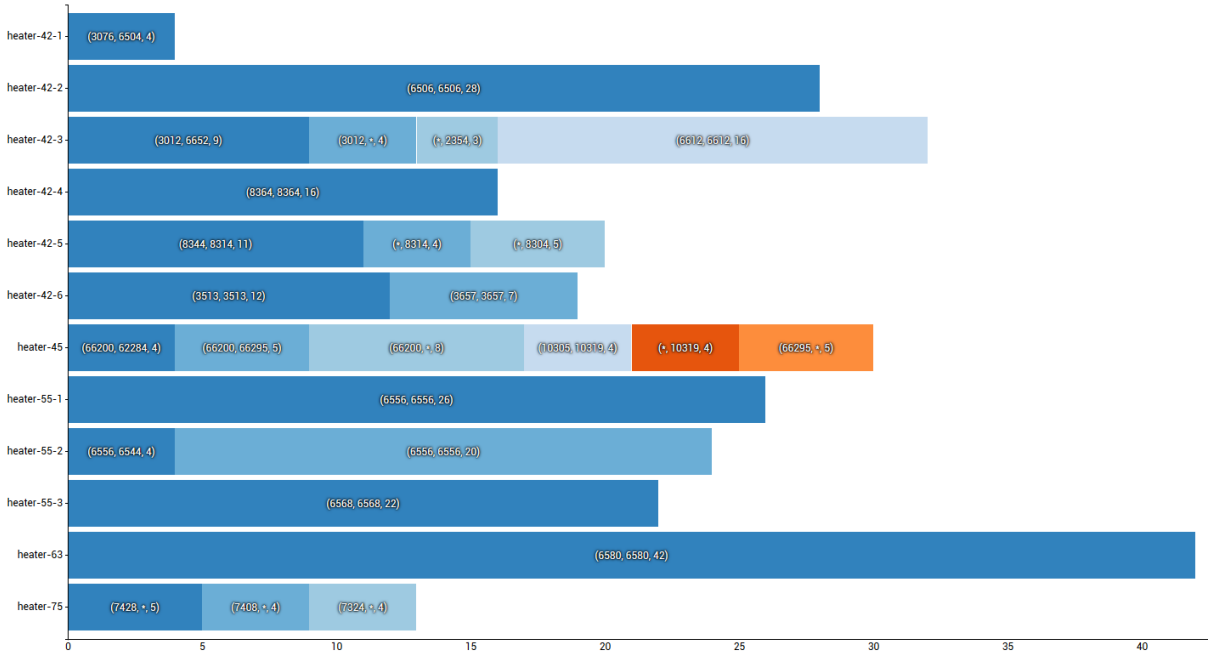


Figura 5.11: Planificación de vulcanizado de FunsaCoop ajustada

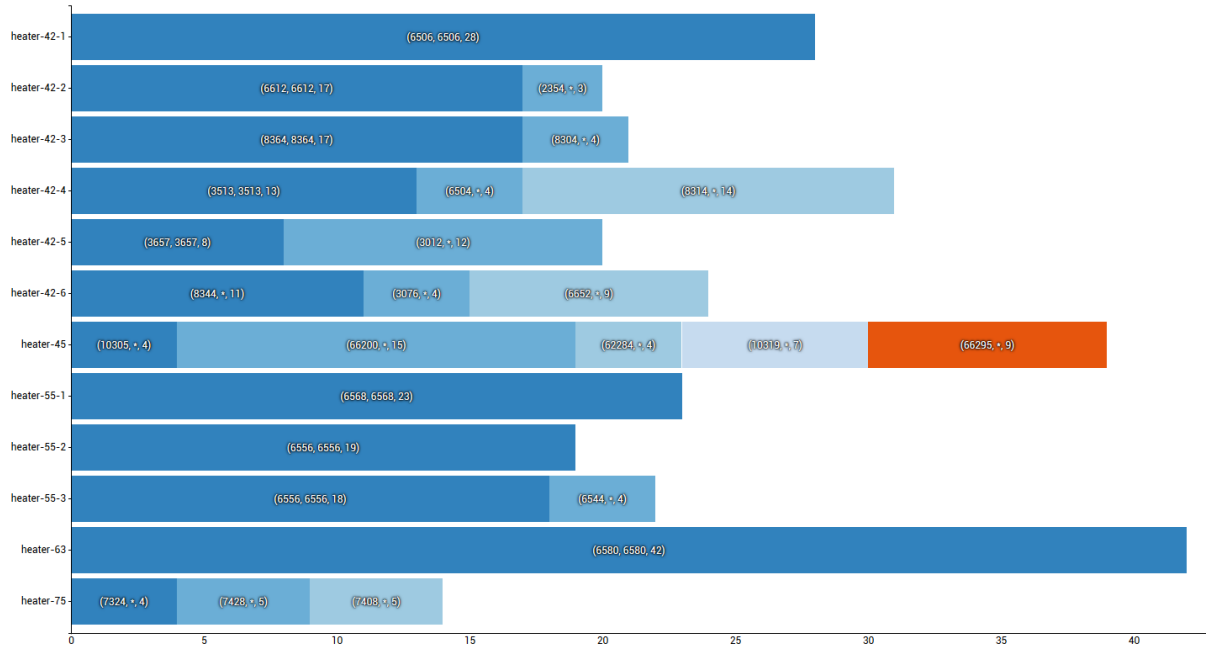


Figura 5.12: Planificación generada por Estimador (H4+FM)

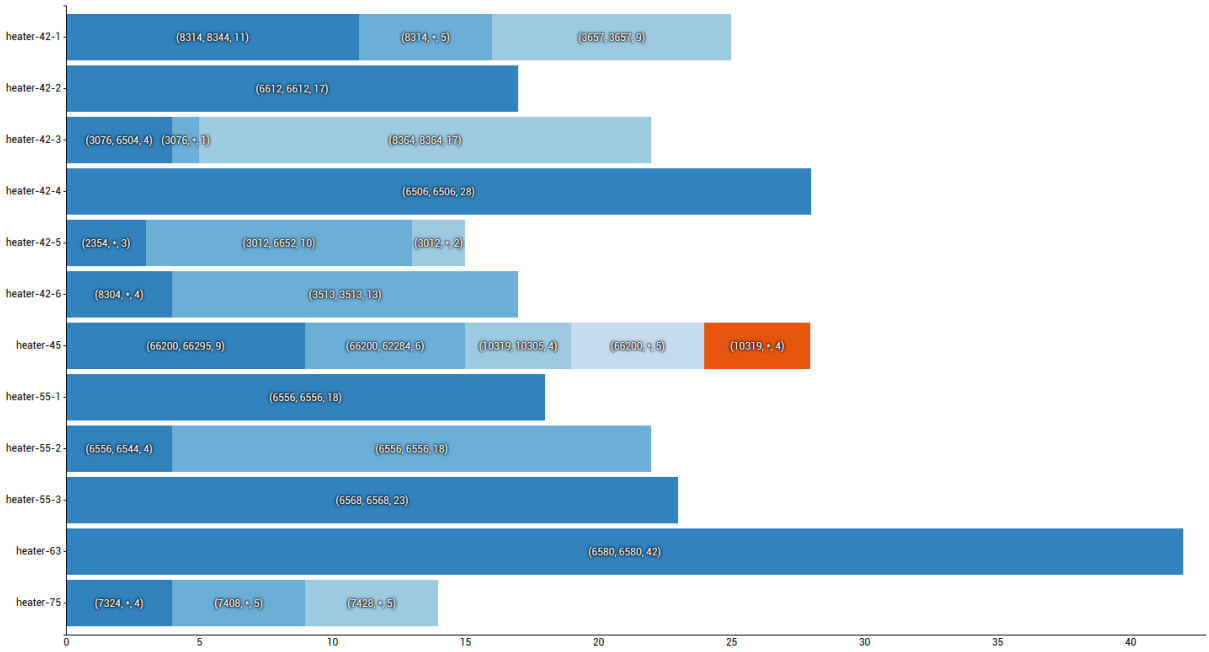


Figura 5.13: Otra planificación generada por Estimador (H4+FM)

CAPÍTULO 6

Conclusiones y trabajos a futuro

Este proyecto de grado estudió el problema de la planificación de la producción en el proceso de vulcanizado de neumáticos, en el contexto de una actividad de extensión para la cooperativa FunsaCoop. Como resultado se elaboraron un conjunto de procedimientos de resolución para un problema de vulcanizado en un ambiente donde coexisten heaters, moldes, demandas, restricciones de capacidad de producción y disponibilidad de piezas compartidas. De la revisión de la literatura realizada, se encontró que varios trabajos han tratado problemas similares, ya que por sus características se requiere gran coordinación entre sus componentes para aprovechar de mejor manera los recursos y tiempo disponibles. La realidad abordada aquí difiere de los demás trabajos dado que se adiciona complejidad a través de restricciones de compatibilidad entre moldes, moldes y heaters y disponibilidad de piezas compartidas.

Se propuso un modelo matemático para el problema tratado y basado en modelos de la literatura existente. El modelo Discrete Lot-Sizing and Scheduling Problem (DSLP) fue utilizado ya que se entendió que es el más adecuado para abarcar la mayoría de las características del problema de vulcanizado de neumáticos. Como función objetivo se buscó minimizar el tiempo total de producción, lo que a su vez minimiza el uso de la caldera a leña utilizada para alimentar los heaters, reduce el desgaste de la maquinaria utilizada, así como la emisión de dióxido de carbono a la atmósfera. En el contexto de FunsaCoop se vio que la formulación modela ciertos aspectos de la realidad y que otros son ignorados, como la distribución y ubicación de piezas en la planta, cantidad de trabajadores, entro otros. También se vio como el horizonte de planificación (parámetro THB) impacta significativamente en los tiempos de resolución si el valor utilizado difiere mucho de la solución óptima.

Se desarrollaron tres formas de resolver el problema de vulcanizado: la aplicación de métodos exactos para resolver la formulación matemática, un método heurístico inspirado en el funcionamiento de la meta-heurística GRASP y un procedimiento híbrido basado en la combinación de los dos anteriores. Para el procedimiento exacto la formulación matemática fue codificada en AMPL y resuelta a través del software comercial CPLEX. Para ello, fue necesario estimar un valor del horizonte de planificación para asegurar la existencia de soluciones. En la experimentación numérica se pudo ver que este estimador sencillo (ecuación 4.1) no era bueno aplicarlo en instancias medianas y grandes debido al tiempo de cómputo requerido, en términos prácticos no es adecuado. De los 110 casos definidos solo fue posible obtener la solución óptima en 44 % de ellos.

Considerando esta desventaja del método exacto, se decidió desarrollar otro procedimiento de resolución mediante técnicas heurísticas. Para lograr tal objetivo, en una primer instancia se elaboró un framework inspirado en la meta-heurística GRASP. El framework está parametrizado por un conjunto de heurísticas constructoras de soluciones y opcionalmente por una rutina de mejora que aplica a cada una de las soluciones generadas. Esta herramienta otorga la posibilidad de ampliar la capacidad de explorar el espacio de búsqueda al poder adicionar fácilmente más heurísticas constructivas. Incluso, puede ser aplicado a problemas similares donde exista el procesamiento simultaneo de diferentes productos, como ser la fabricación de artículos de cerámica, de vidrio o vestimenta.

Al procedimiento de resolución implementado mediante este framework para el problema de vulcanizado se le denominó .Estimador. está conformado por tres heurísticas constructivas (HCTV, HCA y HCD) más un procedimiento de mejora. En términos de efectividad, la forma en que las heurísticas constructivas se complementan permitió que de las 110 instancias definidas, 89 de ellas correspondan a soluciones óptimas obtenidas en menos de 2 segundos. En los casos de estrés, donde la demanda ronda en los 6 millones por cada tipo de neumático, el

tiempo de ejecución para obtener una solución no superaron los 20 segundos. En cuanto al proceso de mejora, cabe destacar que se pudieron mejorar las soluciones dadas por las heurísticas constructivas hasta en un 17 % del valor objetivo, lo cual implica una reducción del tiempo de producción de un 17 %. Puesto que muchas de las instancias probadas corresponden a casos reales de FunsaCoop, se puede concluir que el procedimiento Estimador es capaz de generar soluciones de muy buena calidad.

Por último y con el fin de combinar la fortaleza de los métodos exactos y heurísticos, se elaboró un tercer procedimiento de resolución, basado en la combinación de ambos, que denominamos procedimiento híbrido. Las soluciones dadas por Estimador son adecuadas para utilizarlos como horizonte de planificación (parámetro THB). Esto permitió reducir el tamaño del modelo notoriamente y a CPLEX resolver casos en menor tiempo. Los resultados indicaron una reducción del tiempo de ejecución y aumento de la cantidad de soluciones óptimas. En la experimentación se obtuvo que combinando Estimador y CPLEX 100 soluciones óptimas de las 110 definidas, comparado con las 49 si solo se utilizaba CPLEX.

A continuación se presentan posibles mejoras y puntos para reprofundizar en lo investigado en este proyecto de grado. Respecto al modelo del problema, sería útil generalizar la capacidad de alojamientos de moldes en los heaters. Esto significa no limitar a que todos los heaters tenga solo dos compartimientos, dado que se obtendría un modelo más genérico y capaz de aplicarse en problemas similares de la industria. En cuanto al procedimiento Estimador, las pruebas realizadas deben ser analizadas detenidamente para determinar adecuadamente un valor para la cantidad de iteraciones. Esto puede generar posibles mejoras en tiempos de ejecución, si se encuentra que se puede converger a una solución de buena calidad en menos iteraciones. Otra mejora posible al Estimador sería agregar otras heurísticas constructivas para mejorar y/o ampliar el espacio de búsqueda con el fin de aumentar la posibilidad de obtener mejores soluciones. En cuanto a aspectos técnicos, se propone explorar librerías con estructuras de datos eficientes para mejorar iteraciones y acceso a datos del algoritmo implementado. Si bien se prestó atención en las estructuras de datos utilizadas, existen una gran variedad de ellas optimizadas para usos específicos. También sería interesante explorar otras formas de combinar CPLEX y Estimador. En este proyecto estos dos procedimientos fueron combinados tal que uno utiliza el valor calculado por el otro. Una alternativa posible es elaborar una heurística constructiva para Estimador que aplique técnicas exactas. Por último sería bueno comprender que sucede en instancias donde se utiliza el parámetro THB cercano a la solución óptima ya que por intuición deberían demorar menos por ser una instancia más chica. Específicamente revisar que sucede en los casos que el solver CPLEX toma un tiempo prolongado para resolver ciertas instancias.

Bibliografía

[1] Z. Degraeve and L. Schrage, "A Tire Production Scheduling System for Bridgestone/Firestone Off-The-Road," *Operations Research*, vol. 45, no. 6, pp. 789–796, 1997.

[2] R. Jans and Z. Degraeve, "An industrial extension of the discrete lot-sizing and scheduling problem," *IIE Transactions (Institute of Industrial Engineers)*, vol. 36, no. 1, pp. 47–58, 2004.

[3] S. Yu, D. Yang, X. Wang, K. Zhu, and B. Zheng, "A Two-stage Heuristic Method for Vulcanization Production Scheduling," in *2011 Chinese Control and Decision Conference*, (Mianyang, China), pp. 3601–3605, 2011.

[4] Bridgestone - Firestone Argentina, "Fabricación de neumáticos."

[5] Rockwell Automation, "Tire Curing Press Machine: Streamline Press Performance; Enhance Throughput, Quality and Flexibility," 2014.

[6] C. Fragassa and M. Ippoliti, "Technology assessment of tire mould cleaning systems and quality finishing," *International Journal for Quality Research*, vol. 10, pp. 523–546, 2016.

[7] TireNews, "Tire curing-energy management," 2016.

[8] C. Gicquel, M. Minoux, and Y. Dallery, *Capacitated Lot Sizing models: a literature review.* PhD thesis, ISIMA, 2008.

[9] J. Blazewicz, W. Domschke, and E. Pesch, "The job shop scheduling problem: Conventional and new solution techniques," *European Journal of Operational Research*, vol. 93, no. 1, pp. 1–33, 1996.

[10] P. P. Lukaszewicz, *Metaheuristics for Job Shop Scheduling Problem.* PhD thesis, Aarhus School of Business, 2005.

[11] S. F. Smith, "Is Scheduling a Solved Problem?," in *Multidisciplinary Scheduling: Theory and Applications*, p. 16, Boston, MA: Springer US, 2005.

[12] J. M. Framinan, R. Leisten, and R. Ruiz Garcia, *Manufacturing Scheduling Systems.* Springer-Verlag London, 2014.

[13] Vaivasuata, "Diferencia entre eficacia y eficiencia." http://diferenciaentre.info/diferencia-entre-eficacia-y-eficiencia/, 2014.

[14] A. R. T. Review, O. F. Job, S. Scheduling, T. Anant, S. Jain, S. Meeran, and A. Physics, *A state of the art review of Job-Shop Scheduling Techniques.* PhD thesis, University of Dundee, Dundee, Scotland, 1998.

[15] M. S. Fox and N. Sadeh, "Why is scheduling difficult? A CSP perspective," in *9th European Conference on Artificial Intelligence*, no. June, pp. 754–767, 1990.

[16] J. Lenstra, A. R. Kan, and P. Brucker, *Complexity of machine scheduling problems.* Elsevier, first ed., 1998.

[17] R. Kuik, M. Salomon, and L. N. van Wassenhove, "Batching decisions: structure and models," *European Journal of Operational Research*, vol. 75, no. 2, pp. 243–263, 1994.

[18] C. Gicquel, *MIP models and exact methods for the Discrete Lot-Sizing and Scheduling Problem with sequence-dependent changeover costs and times.* PhD thesis, ISIMA, 2008.

[19] F. Seeanner, "Multi-level lot-sizing and scheduling," in *Multi-Stage Simultaneous Lot-Sizing and Scheduling*, no. 2011, ch. 3, Springer, 1 ed., 2013.

[20] M. Mohammadi, *Metaheuristic Algorithms for Solving Lot- Sizing and Scheduling Problems in Single and Multi-Plant Environments Maryam Mohammadi Faculty of Engineering.* PhD thesis, Faculty of Engineering University of Malaya Kuala Lumpur, 2015.

[21] B. Fleischmann, "The discrete lot-sizing and scheduling problem," *European Journal of Operational Research*, vol. 44, no. 3, pp. 337–348, 1990.

[22] A. Duarte and V. Carvalho, "Discrete lot sizing and scheduling on parallel machines: description of a column generation approach," *Congresso da Associação Portuguesa de Investigação Operacional. Bragança*, pp. 126–134, 2013.

[23] L. Buschkühl, F. Sahling, S. Helber, and H. Tempelmeier, "Dynamic capacitated lot-sizing problems : a classification and review of solution approaches," *OR Spectrum*, vol. 32, pp. 231–261, 2010.

[24] D. Cattrysse, M. Salomon, R. Kuik, L. N. V. Wassenhove, D. Cattrysse, M. Salomon, R. Kuik, and L. N. V. Wassenhove, "A Dual Ascent and Column Generation Heuristic for the Discrete Lotsizing and Scheduling Problem with Setup Times," *Manage. Sci*, vol. 39, no. 4, pp. 477–486, 1993.

[25] S. van Hoesel and A. Kolen, "A linear description of the discrete lot-sizing and scheduling problem," *European Journal of Operational Research*, vol. 75, pp. 342–353, 1994.

[26] R. Jans and Z. Degraeve, "Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1855–1875, 2007.

[27] F. Carvalho, A. S. Pereira, P. Pedroso, and M. Constantino, "Tabu Search for a Discrete Lot Sizing and Scheduling Problem," in *4th Metaheuristics International Conference*, (Porto, Portugal), pp. 697–702, ResearchGate, 2001.

[28] Y.-F. Hung and Y.-C. Hu, "Solving mixed integer programming production planning problems with setups by shadow price information," *Computers & operations research*, vol. 25, no. 12, pp. 1027–1042, 1998.

[29] A. Y. Hung, C. Shih, C. Chen, Y.-f. Hung, C.-c. Shih, and C.-p. Chen, "Evolutionary Algorithms for Production Planning Problems with Setup Decisions," *The Journal of the Operational Research Society*, vol. 50, no. 8, pp. 857–866, 1999.

[30] G. A. Süer, F. Badurdeen, and N. Dissanayake, "Capacitated lot sizing by using multi-chromosome crossover strategy," *Journal of Intelligent Manufacturing*, vol. 19, no. 3, pp. 273–282, 2008.

[31] S. Yu, D. Yang, and K. Zhu, "Production planning simulating system for tire vulcanization based on heuristic algorithm," in *2012 24th Chinese Control and Decision Conference (CCDC)*, (Taiyuan, China), pp. 2661–2666, IEEE, 2012.

[32] H. Piroozfard, K. Y. Wong, and A. Hassan, "A hybrid genetic algorithm with a knowledge-based operator for solving the job shop scheduling problems," *Hindawi Publishing Corporation Journal of Optimization*, pp. 1–14, 2015.

[33] L. Sun, X. Y. Wang, and Y. C. Zhang, "Heuristics algorithms for job-shop scheduling problem: Critical element analysis," in *2012 International Conference on Computer Science and Service System*, (Nanjing, China), pp. 94–97, IEEE, 2012.

[34] M. Seda, "Mathematical Models of Flow Shop and Job Shop Scheduling Problems," *International Journal of Mathematical, Computational, Physical, Electrical and Computer Engineering*, vol. 1, no. 7, pp. 122–127, 2007.

[35] M. Dave and K. Choudhary, "Job Shop Scheduling Algorithms- A Shift from Traditional Techniques to Non-Yraditional Techniques," in *InternationalConference on Computing for SustainableGlobal Development (INDIACom)*, pp. 169–173, New Delhi, India: IEEE, 2016.

[36] R. Zhang, "An Artificial Bee Colony Algorithm Based on Problem Data Properties for Scheduling Job Shops," *Procedia Engineering*, vol. 23, pp. 131–136, 2011.

[37] J. Liangxiao and D. Zhongjum, "An Improved Genetic Algorithm for Flexible Job Shop Scheduling Problem," in *International Conference on Mechatronics, Control and Electronic Engineering (MCE 2014)*, vol. 8, pp. 19–31, 2014.

[38] J. F. Muth and G. L. Thompson, *Industrial scheduling.* Englewood Cliffs, N.J. : Prentice-Hall, 1963.

[39] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management science*, vol. 34, no. 3, pp. 391—-401, 1988.

[40] J. E. Biegel and J. J. Davern, "Genetic algorithms and job shop scheduling," *Computers & Industrial Engineering*, vol. 19, no. 1-4, pp. 81–91, 1990.

[41] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Computers & operations research*, vol. 22, no. 1, pp. 5–13, 1995.

[42] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, p. 2005, 2005.

[43] J.-H. Yang, L. Sun, H. P. Lee, Y. Qian, and Y.-c. Liang, "Clonal Selection Based Memetic Algorithm for Job Shop Scheduling Problems," *Journal of Bionic Engineering*, vol. 5, no. 2, pp. 111–119, 2008.

[44] S. Fernandes and H. Lourenco, "A Simple Optimised Search Heuristic for the Job Shop Scheduling Problem," in *Recent Advances in Evolutionary Computation for Combinatorial Optimization*, pp. 203—-218, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[45] D. Y. Sha and H.-H. Lin, "A multi-objective PSO for job-shop scheduling problems," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1065–1070, 2010.

[46] R. Zhang, "An artificial bee colony algorithm based on problem data properties for scheduling job shops," *Procedia Engineering*, vol. 23, pp. 131–136, 2011.

[47] W. Wisittipanich and V. Kachitvichyanukul, "Two enhanced differential evolution algorithms for job shop scheduling problems," *International Journal of Production Research*, vol. 50, no. 10, pp. 2757–2773, 2012.

[48] M. Ziaee, "Job shop scheduling with makespan objective: A heuristic approach," *International Journal of Industrial Engineering Computations*, vol. 5, no. 2, pp. 273–280, 2014.

[49] Y. Long Liu, P. C. Ma, and F. Tao, "A hybrid particle swarm optimization and simulated annealing algorithm for job-shop scheduling," in *2014 IEEE International Conference on Automation Science and Engineering (CASE)*, (Taipei, Taiwan), pp. 125–130, IEEE, 2014.

[50] A. Somani and D. P. Singh, "Parallel Genetic Algorithm for solving Job-Shop Scheduling Problem Using Topological sort," in *Advances in Engineering and Technology Research (ICAETR), 2014 International Conference on*, pp. 1–8, IEEE, 2014.

[51] W. Cheung and H. Zhou, "Using Genetic Algorithms and Heuristics for Job Shop Scheduling with Sequence-Dependent Setup Times," *Annals of Operations Research*, vol. 107, no. 1-4, pp. 65–81, 2001.

[52] H. Lourenco and J. Noivo, *Solving Two Production Scheduling Problems with Sequence-Dependent Set-Up Times*. PhD thesis, Universitat Pompeu Fabra, 1999.

[53] R. Moghaddas and M. Houshmand, "Job-shop scheduling problem with sequence dependent setup times," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. II, (Hong Kong), pp. 19–21, 2008.

[54] P. Sharma and A. Jain, "A review on job shop scheduling with setup times," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 230, no. 3, pp. 517–533, 2015.

[55] C. R. Vela, R. Varela, and M. A. González, "Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times," *Journal of Heuristics*, vol. 16, no. 2, pp. 139–165, 2010.

[56] R. Varela, D. Serrano, and M. Sierra, *New Codification Schemas for Scheduling with Genetic Algorithms*, pp. 11–20. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[57] M. Tiedemann, *Algorithmic Approaches to Flexible Job Shop Scheduling Master Thesis*. PhD thesis, Faculty of Mathematics and Computer Science, 2012.

[58] B. Yao, C. Yang, J. Hu, J. Yao, and J. Sun, "An improved ant colony optimization for flexible job shop scheduling problems," in *Sci. Program.*, vol. 2017, (London, UK, United Kingdom), pp. 2127–2131, Hindawi Limited, 2017.

[59] P. Wojakowski and D. Warzolek, "Research Study of State-of-the-Art Algorithms for Flexible Job-Shop Scheduling Problem," in *Technical Transactions*, pp. 381—-388, 2013.

[60] D. Cinar, Y. I. Topcu, and J. A. Oliveira, "A Taxonomy for the Flexible Job Shop Scheduling Problem," in *Optimization, Control, and Applications in the Information Age* (A. Karakitsiou and A. Migdalas, eds.), vol. 130, (Cham), pp. 17–37, Springer International Publishing Switzerland, 2015.

[61] I. Kacem, "Genetic Algorithm for the Flexible Jobshop Scheduling Problem," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3464–3469, Researchgate, 2003.

[62] I. Kacem and A. El Kamel, "Optimization by Phases for the Flexible Job-Shop Scheduling Problem," *5th Asian Control Conference*, pp. 1889–1895, 2004.

[63] H. Liu, A. Abraham, and C. Grosan, "A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems," in *2007 2nd International Conference on Digital Information Management*, vol. 1, (Lyon, France), pp. 138–145, IEEE, 2007.

[64] S. Y. S. Yang, Z. G. Z. Guohui, G. L. G. Liang, and Y. K. Y. Kun, "A novel initialization method for solving Flexible Job-shop Scheduling Problem," in *2009 International Conference on Computers Industrial Engineering*, pp. 68–73, Troyes, France: IEEE, 2009.

[65] L. Di and T. Ze, "A Genetic Algorithm with Tabu Search for Multi-Objective Scheduling Constrained Flexible Job Shop," in *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*, (Harbin, China), pp. 1662–1665, IEEE, 2011.

[66] X. Liang, S. Weiping, and M. Huang, "Flexible job shop scheduling based on multi-population genetic-variable neighborhood search algorithm," *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, vol. 01, no. Iccsnt, pp. 244–248, 2015.

[67] J.-Q. Li, Q.-K. Pan, P. N. Suganthan, and T. J. Chua, "A hybrid tabu search algorithm with an efficient neighborhood structure for the flexible job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 52, no. 5, pp. 683–697, 2011.

[68] A. Jamili, M. A. Shafia, and R. Tavakkoli-Moghaddam, "A hybrid algorithm based on particle swarm optimization and simulated annealing for a periodic job shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 1, pp. 309–322, 2011.

[69] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Computers and Operations Research*, vol. 35, no. 9, pp. 2892–2907, 2008.

[70] N. Al-Hinai and T. Y. ElMekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *International Journal of Production Economics*, vol. 132, no. 2, pp. 279–291, 2011.

[71] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Computers and Industrial Engineering*, vol. 56, no. 4, pp. 1309–1318, 2009.

[72] W. Teekeng, A. Thammano, P. Unkaw, and J. Kiatwuthiamorn, "A new algorithm for flexible job-shop scheduling problem based on particle swarm optimization," *Artificial Life and Robotics*, vol. 21, no. 1, pp. 18–23, 2016.

[73] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong, "A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems," *Applied Soft Computing*, vol. 10, no. 3, pp. 888–896, 2010.

[74] M. I. Hosny, *Investigating Heuristic and Meta-Heuristic Algorithms for Solving Pickup and Delivery Problems*. PhD thesis, Cardiff University, 2010.

[75] D. S. Garey, Michael R. and Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman \& Co., first ed., 1979.

[76] V. Iv and M. M. Gómez, "Las metaheurísticas : tendencias actuales y su aplicabilidad en la ergonomía.," *12*, vol. IV, pp. 108–120, 2014.

[77] Á. G. Sánchez, "Técnicas metaheurísticas," 2013.

[78] X.-f. Zhang, "Combining PSO and local search to solve scheduling problems," in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, pp. 347–354, New York, NY, USA: ACM, 2011.

[79] J. Tang, G. Zhang, B. Lin, and B. Zhang, "A Hybrid Algorithm for Flexible Job-shop Scheduling Problem," *Procedia Engineering*, vol. 15, pp. 3678–3683, 2011.

[80] D.-w. Huang and J. Lin, "Scaling Populations of a Genetic Algorithm for Job Shop Scheduling Problems using MapReduce," in *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science*, pp. 780 – 785, IEEE Xplore, 2010.

[81] J. F. Goncalves, "A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem," *European Journal of Operational Research*, vol. 167, no. September 2002, pp. 77 – 95, 2002.

[82] R. W. Eglese, "Simulated annealing: A tool for operational research," *European Journal of Operational Research*, vol. 46, no. 3, pp. 271–281, 1990.

[83] E. D, "Parallel Taboo Search Techniques for the Job Shop Scheduling Problem," *ORSA Journal on Computing*, vol. 6, no. 2, pp. 1—-10, 1992.

[84] K. Schmidt, *Using tabu-search to solve the job-shop scheduling problem with sequence dependent setup times*. PhD thesis, Brown University, USA, 2001.

[85] J. E. Fuster, *Eficiencia Energética y Robustez en Problemas de Scheduling*. PhD thesis, Universitat Politècnica de València, 2016.

[86] M. G. Resende and C. C. Ribeiro, "Greedy Randomized Adaptive Search Procedures," in *Handbook of Metaheuristics*, vol. 146, pp. 219—-249, Boston, MA: Springer US, 1994.

[87] M. L. Pinedo, *Scheduling: Theory,Algorithms, and Systems*. NewYork: Springer, third ed., 2008.

[88] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Springer US, first ed., 2010.

[89] J. M. G. Nieto, *Algoritmos Basados en Cúmulos de Partículas Para la Resolución de Problemas Complejos*. PhD thesis, Universidad de Málaga, 2006.

ANEXO A

Estado del arte

Introducción

El objetivo de este documento es hacer una puesta a punto respecto a los avances en los problemas Discrete Lot-Sizing y Job-Shop Scheduling en un contexto de manufactura y planificación de la producción. Ambos problemas son modelos destinados a crear un ordenamiento de tareas bajo ciertas restricciones como capacidad de producción, recursos y maquinaria. El ordenamiento de tareas conocido también como scheduling, es una actividad que de forma consiente o no, se aplica todos los días, por más sencillas resulten ser se caracterizan por ser tareas de la forma "decidir-hacer". A nivel industrial el ordenamiento es un proceso más complejo, donde la transformación de materia prima a un producto implica la ejecución de una sucesión de tareas respectando la asignación de recursos, de un orden y maquinaria [9]. La forma en que se define el proceso puede generar un mal producto final, más tiempo requerido, mala experiencia de compra, mal uso de recursos energéticos y por supuesto, elevación de costos. Adicionando el hecho de formar parte de un mundo globalizado, donde los gobiernos y empresas buscan el control del consumo energético, de los precios del combustible, la competencia en el mercado y del preocupante calentamiento global, es suficiente para generar incentivos y obligación en la búsqueda de nuevas formas que generen procedimientos más eficaces [10]. La aplicación de medidas de eficiencia energética son la mejor manera de reducir el daño al medio ambiente, pero también puede incrementar la rentabilidad del negocio. Debido a esto, el estudio de la obtención de soluciones optimas que brinden seguridad, robustez y eficiencia ha generado un gran intereses en los últimos 50 años [10].

El estudio persistente a lo largo de los años ha generado un avance tecnológico fortaleciendo [11]:

- Escalabilidad: las técnicas actuales de ordenamiento son capaces de resolver grandes problemas en tiempo razonable.
- Flexibilidad en modelado y representación de problemas: las técnicas actuales permiten generar ordenamientos bajo un conjunto diverso de restricciones y capacidades de recursos.
- Optimización: el desarrollo de técnicas de búsqueda de óptimos globales, locales y desarrollo de metaheurísticas han ampliado las ventajas en la optimización de ordenamientos, permitiendo además la integración de otras técnicas y modelos matemáticos (ej. Inteligencia Artificial, Programación Entera, etc.).

Es importante resaltar que las los avances en las técnicas no son suficientes para hacer rentable un proceso; existen influencias y decisiones externas como la preferencias de usuarios a la hora de elegir un producto, el tipo de materia prima utilizada, los precios o la reputación de una marca que determinan implícitamente el problema a encarar.

El resto de este documento se organiza como sigue. En la Sección 2 introduce al problema de ordenamiento, incluyendo definiciones, estrategias de decisión y clasificaciones. En la Sección 3 se realiza un relevamiento al problema Discrete Lot-Sizing, abarcando definiciones, formulación, revisión de literatura, técnicas de resolución y análisis en aplicaciones industriales. De igual modo, en la Sección 4 se trata el tema Job-Shop Scheduling. Finalmente dn la Sección 4 se exponente las conclusiones.

Se recomienda al lector como complemento el Anexo donde se describe avances en algoritmia para la resolución de problemas de ordenamiento.

Problema de ordenamiento

Introducción

El proceso de administración y planificación industrial está formado por un conjunto amplio de decisiones realizadas durante un periodo de tiempo, teniendo como objetivo asegurar reducción de costos, aumento en calidad y el empleo de menos tiempo de producción. Estás decisiones impactan en la compañía y a su entorno. Tradicionalmente este conjunto de decisiones han sido descompuestas en forma de estructura jerárquica (ver figura A.1); desde arriba hacia abajo el nivel de detalle y dificultad incrementa.

El proceso de decisión comienza en el tope de la estructura, dando lugar a las Decisiones estratégicas; estas son de carácter macro y tienen impacto a largo plazo; la ubicación de la planta, distribución de los sectores y diseño y el manejo con los proveedores son algunos ejemplos de decisión.

Una vez definida la estructura base se procede a tomar más decisiones respecto a cual es la mejor forma de aprovechar los recursos disponibles, generalmente se clasifican en Decisiones tácticas (mediano plazo) y operativas (corto plazo). Estos tipos de decisiones se relacionan con la complexidad del proceso de producción y la calidad de información para tomar una decisión.

Al momento de planificar el uso de recursos para satisfacer una demanda, no siempre se cuenta con la información necesaria que conduce todo el proceso de producción, por lo que es necesario realizar una estimación utilizando datos históricos o casuales. Para esto se realiza un Plan Agregado de Producción que consiste en generar una estimación basada en otras de productos con similares características y/o comportamiento

donde la entrada es la estimación de un grupo de productos con similares características y/o comportamiento respecto a la demanda que tiene que satisfacerse sobre un periodo de tiempo semanal a mensual. Este plan da como resultado una ayuda para estimar los recursos a utilizar, la cantidad de material, etc., pero no da indicaciones del orden de producción que se debe aplicar. Posteriormente este plan se utiliza cuando es conocida la demanda a satisfacer y esta es justamente la misión de un ordenamiento: especificar que trabajo debe utilizar cada recurso y cuando[12, p. 4-5].

En la industria manufacturera el ordenamiento es una herramienta utilizada para obtener eficiencia y eficacia en utilización de recursos. Eficiencia refiere al uso óptimo y adecuado de los recursos, es lograr el objetivo con lo que se tiene; eficacia apunta al logro de los objetivos, se tiene en cuenta los resultados, no el proceso que se llevó a cabo para llegar a los mismos. La eficiencia se refiere a hacer las cosas bien, es obtener el mejor o máximo rendimiento utilizando un mínimo de recursos. La eficacia, por otra parte, es hacer las cosas de la manera correcta para así alcanzar el resultado deseado. La eficiencia se centra en el proceso que se sigue para lograr algo, tomando en cuenta los "medios"; mientras que la eficacia se centra en el logro o alcance final, es decir en los "fines" [13].

Pese a que su uso genera beneficios, muchos sectores planifican su producción de forma manual, basándose en la experiencia adquirida. Por ejemplo la fabrica de neumáticos para camiones, tractores y excavadoras Bridgestone/Firestone Off-The-Road (BFOR) de Estados Unidos [1] mantiene el ordenamiento de producción ejecutado por un servicio de call center, encargado de atender los pedidos y administrar un inventario de neumáticos. A pesar de que la experiencia del comprador es buena (no debe esperar por todo el proceso de producción de una rueda) el problema radica en no producir más ruedas de las demandadas.

Es importante plantearse "¿porqué invertir tiempo y dinero en investigar y desarrollar metodologías para resolver este problema ?" ¿porqué no solo contar con los años de experiencia de los funcionarios y sectores similares en la industria?

En primer lugar, han emergido un amplio número de sectores de productos y servicios que deben lidiar con este problema y sus consecuencias, algunos ejemplos son el Sector de transporte, sector informático, sector administrativo, industria papelera e industria de comidas.

En segundo lugar, todos estos problemas y otros pertenecen a la categoría de *Problemas Combinatorios*, es

Figura A.1: Estructura de decisiones

decir, el dominio es un espacio de soluciones muy amplio y se tiene el objetivo de encontrar la mejor solución limitado a los recursos disponibles y sujeto a restricciones; por lo que hacer el trabajo de forma manual puede incrementar más los margenes de error. Para ver la dimensión del problema, existe una clase de scheduling denominada Flow Shop Scheduling y dos de sus parámetros son la cantidad de máquinas disponibles y la de tareas para realizar; si se cuenta con 20 tareas y 10 máquinas, existen $20!^{10}$ posibilidades de ordenarlos, este número es aproximado a $7{,}2651 \times 10^{183}$ y excede a la edad estimada del universo en micro segundos [14, p. 5].

Es claro que recorrer de manera exhaustiva un espacio de soluciones tan inmenso no es práctico y por eso que el estudio de la optimización combinatoria, con comienzos a mitad del siglo XX hasta hoy en día es una rama ampliamente estudiada por su complejidad y aplicaciones.

A pesar de que existen varios tipos de ordenamientos (scheduling) con sus particularidades, el problema base consiste en organizar tareas que junto a un recurso asignado será ejecutada en un intervalo de tiempo; por ende es muy común ver en la literatura y en la práctica modelos base que abstraen un problema para luego ser adaptados a uno específico. Esto implica que la dificultad de encontrar un ordenamiento puede variar, ya sea por el entorno de trabajo de producción, las restricciones, los recursos, decisiones políticas, criterios de optimización, cultura de la organización, etc. En general, un problema de ordenamiento se compone de los siguientes parámetros (en la siguiente sección se trata en más detalle):

- Recursos ó maquinaria.
- Operaciones.
- Restricciones.
- Tareas.
- Criterios de optimización.

Un Recurso o Máquina es el medio para ejecutar una Tarea; puede ser renovable y/o consumible, por ejemplo la cantidad de unidades de transporte en una empresa de correo, las pistas de aterrizaje en los aeropuertos, la cantidad de unidades de procesamiento (CPU) de un sistema de computo.

Por otro lado, una Operación representa un proceso; por ejemplo asignar rutas para repartos de paquetes, asignar pistas de aterrizaje o ejecutar un programa en una PC. Además, a cada operación se le puede asignar Restricciones y puede requerir del uso de más de un Recurso o Máquina. Algunas restricciones comúnmente utilizadas son:

- Tiempo de inicio más temprano: se refiere a que tan anticipada puede ser la ejecución de una operación.
- Tiempo de fin más tardío: es lo más tarde que una operación pueda finalizar.
- Prioridad.
- Precedencia entre otras operaciones: es una relación de orden.

Las restricciones no son elementos que adicionan complejidad a un problemas, pueden ser utilizados como medio para simplificar un problema.

Por otra parte las Tareas encapsulan una funcionalidad o meta por medio de un conjunto de operaciones; algunos ejemplos son entregar la producción de calzado en 6 meses, asignar las jornadas de trabajo de cada funcionario, etc.

Los Criterios de Optimización son factores que determinan la optimalidad del ordenamiento, los más frecuentes en utilización son[15]:

- Makespan: Tiempo entre inicio y final de la ejecución de todas las tareas.
- Lateness: Maximizar el tiempo entre que una operación es completa y su tiempo de finalización. Controla la varianza del tiempo en completar cada operación.
- Costo total.

El uso de algunos criterios de optimización suele generar que otros sean afectados; en el caso de minimizar el uso de recursos o el makespan puede generar que los costos de producción se reduzcan. A pesar que muchos se relacionan con el tiempo, existen pequeñas modificaciones en los modelos que pueden influir en la dificultad de resolución.

Esto es sólo un conjunto reducido de los parámetros posibles que se pueden aplicar a un problema de ordenamiento. Las características especificas del problema determinan la complejidad que se enfrenta.

Clasificación y tipos de ordenamientos

Un problema de ordenamiento se puede definir utilizando la forma de terna $\alpha|\beta|\gamma$, representación propuesta por Graham en 1979. Aquí α representa el ambiente de máquina, β la característica de una operación y γ el criterio de optimización. A continuación se detalla cada parámetro.

Ambiente de máquina α

El ambiente de máquina representa la configuración de la maquinaria en el sistema. Existen dos modelos: single-stage y multi-stage [10, p. 11]; en single-stage cada operación se realiza en una máquina o en un conjunto de máquinas operando en paralelo. En el caso de máquinas paralelas cada una de ellas realiza la misma función y se pueden clasificar de tres formas [16, p. 27]:

- Máquinas paralelas idénticas: el tiempo de procesamiento es independiente de la máquina que realiza una operación.
- Máquinas paralelas uniformes: cada máquina opera a velocidades distintas.
- Máquinas paralelas no relacionadas: el tiempo de procesamiento de cada operación depende de la máquina asignada.

En el modelo multi-stage cada tarea requiere pasar por s máquinas o etapas; en contraparte a single-stage, éstas brindan distintas modalidades. Hay tres tipos principales:

- Flow Shop: cada operación debe visitar a todas las máquinas utilizando el mismo orden de especificación. A pesar de que las operación comparten un mismo orden de visitas a las máquinas, el orden de procesamiento puede variar. Si se cuenta con n operaciones, una máquina puede procesarlos de $n!$ formas; generalizando a m máquinas hay $(n!)^m$ formas en total de realizar todos los trabajos.

- Job Shop: es otro de los modelos de multi-stage y al igual que Flow-Shop se dispone de m máquinas, diferenciando que cada tarea puede tener distintos ordenes de especificación. Por ejemplo si se dispone de tres máquinas y tres operaciones, las instancias $T1 = (m_1, m_2, m_3)$, $T2 = (m_2, m_1, m_3)$ y $T3 = (m_3, m_2, m_1)$ son válidas. La posibilidad de variar los ordenes de procesamiento en las operaciones que componen una tarea permite a este modelo ser más adecuado en la práctica; en términos de resolución implicada mayor complejidad que Flow-Shop, dado que las combinaciones son mayores.

- Open Shop: Este modelo es el más general y complejo que los indicados anteriormente. Se puede decir que Flow-Shop y Job-Shop son casos particulares, la gran diferencia es que no hay un orden de especificación para las tareas. Si se cuenta con las máquinas $m1, m2$ y $m3$ cada operación puede recorrerlas de la forma:

  - $m1, m2, m3$
  - $m1, m3, m2$
  - $m2, m1, m3$
  - $m2, m3, m1$
  - $m3, m1, m2$
  - $m3, m2, m1$

  La libertad que ofrece este modelo implica manejar más combinaciones, incrementando considerablemente el espacio de soluciones.

En las Figuras A.2 y A.3 se muestra el esquema correspondiente a single-stage y multi-stage [10, p.11-12]:

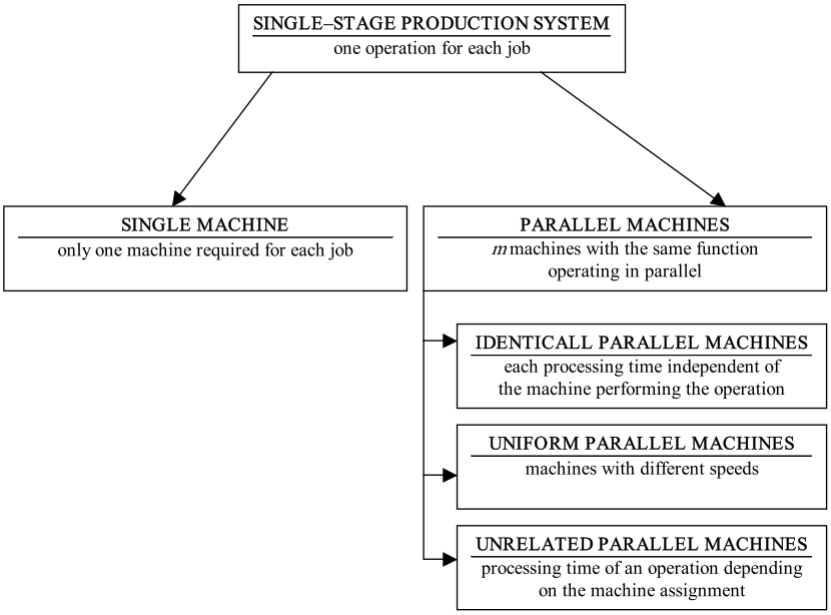


Figura A.2: Ambiente de máquina, producción single-stage (Fuente: **??**)

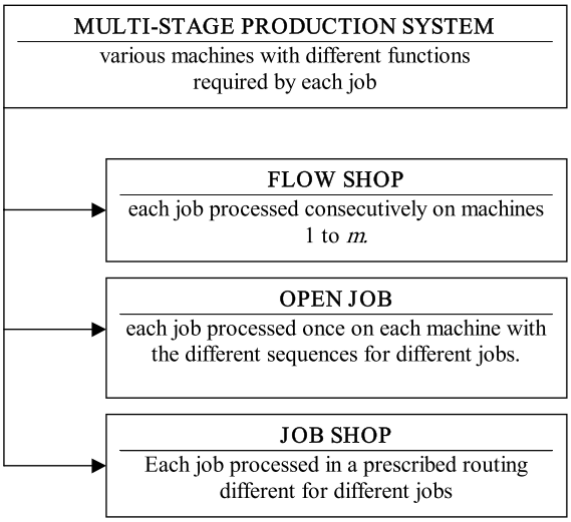


Figura A.3: Ambiente de máquina, producción multi-stage (Fuente: **??**)

Características de una operación (β)

Las características de una operación es otro de los parámetros que determinan un ordenamiento, algunos de ellas son:

- Tiempo de procesamiento: Dada la operación j, su tiempo de procesado puede variar según la configuración de las máquinas. En el caso de single-stage con máquinas paralelas idénticas su tiempo de procesado es p_j; en single-stage con máquinas paralelas uniformes el tiempo de procesado en la máquina i se puede expresar como p_j/s_i, siendo s_i la velocidad de procesado de la máquina i; en single-stage con máquinas paralelas independientes, flow-shop y open-shop el tiempo de procesado en la máquina i es $p_{i,j}$ es; en job-shop su tiempo es $p_{i,j}$, siendo i la operación de la tarea j.
- Disponibilidad: Una operación puede estar restringida por su Release Date que define el instante de tiempo que queda disponible para poder ser procesado; Deadline es otro atributo que define el instante de tiempo limite que un trabajo debe realizarse.
- Tiempo de setup: corresponde al tiempo necesario para ajustar una máquina previamente a su uso.
- Dependencias de operaciones: esto significa que si una operación i depende de la j, esta no puede ser procesada si la operación j no la ha sido.
- Preemption: indica si una operación puede ser interrumpida. En algunos modelos se aloja que el procesamiento de las operaciones sea interrumpido y luego reanudado.

Criterios de optimización (γ)

El último parámetro para definir un ordenamiento es el Criterio de Optimización; consiste en especificar cierta función con el objetivo de minimizar o maximizar, se suele definir a través de varios factores determinados por el problema base. A continuación se listan los más utilizados [16, p. 14]:

- Tiempo máximo de finalización.
- Makespan.
- Máximo retraso.
- Máximo costo.
- Máxima precocidad.
- Tiempo de terminación ponderado total.

- Tiempo de flujo ponderado total.
- Precocidad ponderada total.
- Costo total.

Discrete Lot-Sizing and Scheduling Problem

Introducción

Existe una gran variedad de modelos que resuelven la planificación de producción y administración de stocks. En particular, el modelo Lot-Sizing (LS) definido como "agrupamiento de artefactos para el transporte o proceso de producción al mismo tiempo"[17] busca optimizar la asignación de tiempo y niveles de producción. Estos problemas dan lugar cuando se realiza un proceso de producción que funciona en base a ordenes realizadas por clientes ú otras instalaciones (make-to-order).

Como resultado es necesario mantener el stock producido en inventarios y debido a los costos de oportunidad del capital y los directos en almacenar los bienes se generan pérdidas, por lo tanto deben ser evitados. En otra mano, si diferentes partes de la instalación hacen uso compartido de los recursos, por ejemplo de máquinas y una configuración previa (setup) debe tomar lugar para preparar una operación, el costo de oportunidad es incurrido desde que la producción es retrasada. Por lo cual este modelo debe generar un ordenamiento de tareas capaz de minimizar costos. Entre sus aplicaciones prácticas se abarca la industria de neumáticos, plástica, textil, papelera entre otros [18, p. 13].

Esta sección se enfoca en un modelo de planificación de producción que se sitúa especialmente en los procesos industriales: Discrete Lot-Sizing Problem (DLSP). DLSP es una sub-clase de Lot-Sizing y se basa en la siguientes suposiciones:

- La demanda de los productos es determinista y variable.
- El plan de producción se ejecuta sobre un horizonte de tiempo finito dividido en varios intervalos de poca duración (jornada laboral o día).
- A lo sumo un tipo de artículo puede ser producido por periodo y la instalación procesa un producto a capacidad máxima o queda completamente en estado ocioso. A esto se le conoce como "all-or-nothing policy".
- Setup-cost o changeover: Es el costo generado cuando se cambia la configuración de un recurso para pasar a producir otro tipo de artefacto. En estas tareas se incluye limpieza, pre-calentamiento, ajustes de maquinaria, calibración, pruebas, etc.
- Inventory holding cost: Cada artefacto producido debe ser mantenido en stock hasta que es usado para satisfacer la demanda. Entre los gastos producidos se encuentra el costo asociado al alojamiento físico, impuestos y seguros, el deterioro, roturas y obsolescencia de mercancía.

Para reducir el setup-time, setup-cost y obtener un uso eficiente de los recursos, en la práctica debe producirse en grandes cantidades, en consecuencia los costos de alojamiento incrementan ya que la cantidad ofrecida puede exceder la demandada. Por el contrario, el nivel de inventario se puede mantener bajo si la producción es realizada en pequeñas cantidades pero la frecuencia de setups necesarios es mayor, por lo que aumentan los costos de setup. Por lo tanto, el objetivo de DLSP es encontrar una solución que reduzca los costos de producción y alojamiento en función a la demanda [18, p. 4].

Definiciones y aspectos generales

Para saber como resolver este tipo de problemas, primero se tratará sobre los principales atributos del modelo Lot-Sizing y sus extensiones. La complejidad de estos modelos dependen de las características tomadas acorde al problema a desarrollar. Según [8] una primer clasificación puede darse con las siguientes características:

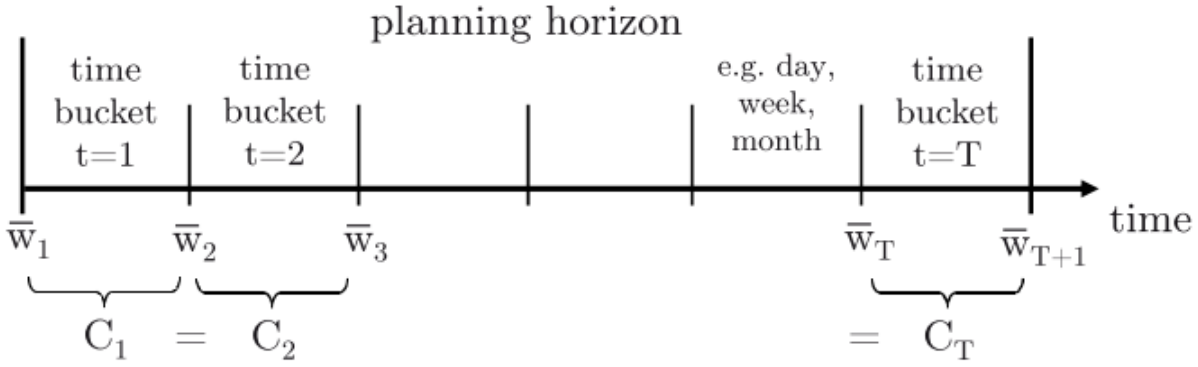


Figura A.4: Períodos de tiempos iguales (Fuente: Multi-level lot-sizing and scheduling [19])

Número de recursos: Los artefactos se pueden fabricar utilizando solo una máquina (single-resource model) o múltiples (multi-resource model). El uso de máquinas paralelas adiciona complejidad debido a que se tiene que asignar lotes de producción a cada una.

Número de niveles (level number): Los sistemas de producción pueden ser single-level o multi-level. Los single-level se caracterizan por requerir de una operación o etapa para obtener un artefacto final; los multi-level requieren de varias etapas, implícitamente se genera una dependencia entre etapas dado que el resultado de una etapa es la entrada de la siguiente.

Capacidad de maquinaria: Un aspecto adicional que influye en el modelo es si la producción es de carácter single-item o multi-item. En el modo single-item cada máquina produce un solo tipo de artefacto y en el primer periodo de producción es necesaria una configuración inicial con un costo denominado startup-cost. En el escenario multi-item las máquinas son capaces de producir varios tipos de artefactos y aparte del startup-cost el cambio de producción de un tipo a otro genera un costo de configuración denominado changeover-cost.

Discretización del horizonte de planificación: Los problemas de Lot-Sizing pueden ser small-bucket y/o big-bucket. En small-bucket el periodo de tiempo es pequeño (jornadas laborales, días) de tal manera que se pueda producir solo un tipo de producto por periodo. Mientras que los big-bucket los periodos de tiempo son largos (semanas, meses) y es posible producir más de un tipo de artefacto por periodo (ver Figuras A.4 y A.5). La discretización es utilizada en modelos donde la demanda es dinámica (por periodo), sin embargo la demanda puede ser continua en el horizonte de planificación (ver Figura A.6).

En base a este criterio se puede encontrar dos grandes grupos de modelos desarrollados. Por un lado están los modelos que aplican una escala continua de tiempo, demanda constante y un horizonte de planificación infinito. En este grupo se encuentra Economic Order Quantity (EOQ) y Economic Lot Scheduling Problem (ELSP). EOQ asume que el proceso producción es single-level y no tiene restricciones de capacidad, por lo que se puede ver como un problema single-item. La demanda de cada artefacto es continua con un rango constante y la linea de tiempo para la planificación es continua e infinita. Por otro lado el modelo Economic Lot Scheduling Problem (ELSP) aplica restricciones de capacidad y adiciona la capacidad de producir más de un artefacto por máquina (multi-item). ELSP mantiene algunas de las características de EOQ: es de single-level, asume la demanda estacionaria, utiliza una linea de tiempo continua y la planificación horizontal es infinita.

En el otro grupo están los modelos que aplican escala de tiempo discreto, demanda dinámica y un horizonte de tiempo finito. Este grupo se referencia por Dynamic Lot Sizing; entre ellos se encuentra el modelo de Wagner-Whitin y en particular Discrete Lot Sizing Problem. Wagner-Whitin[1958] (WW) incorpora las demandas dinámicas: la linea de planificación se considera finita y divida en sub-intervalos denominados periodos. Este modelo no considera la restricciones de capacidad, por lo que es de tipo single-level y single-item.

En la Figura A.7 se puede ver un esquema de la agrupación de los distintos modelos de Lot-Sizing y en las referencias [8],[19] explican en detalles los distintos modelos de la clase Lot-Sizing, detallando sus características y formulación.

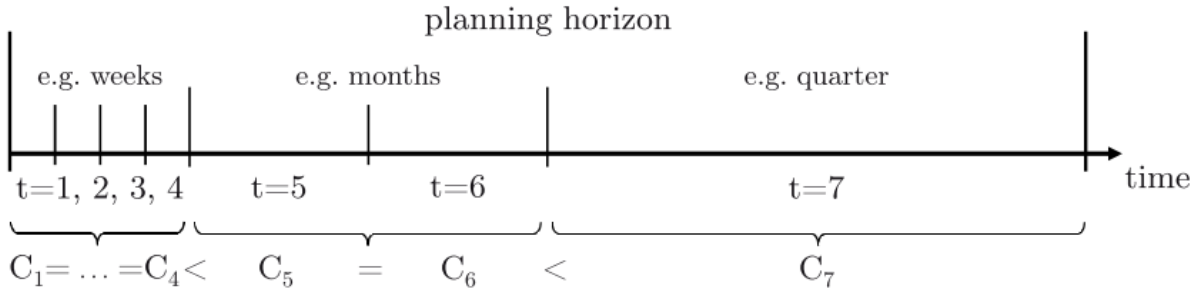
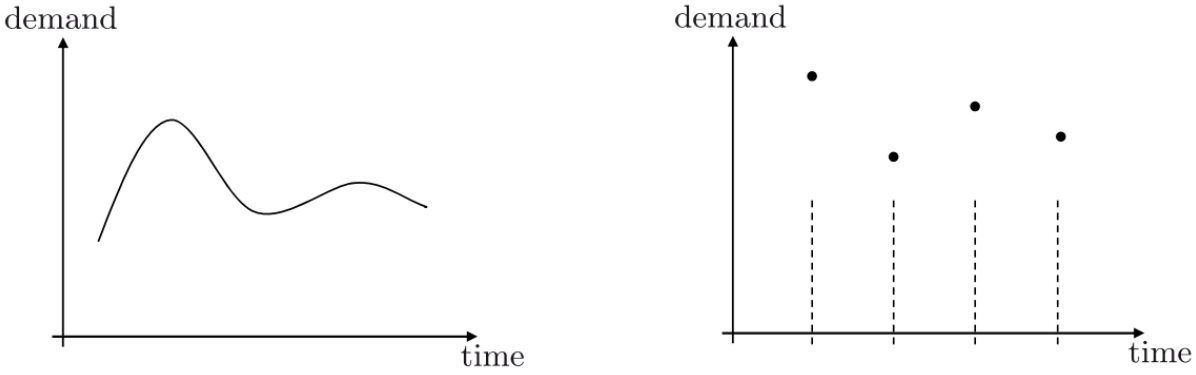


Figura A.5: Períodos de tiempos diferentes (Fuente: Multi-level lot-sizing and scheduling [19])



(a) Continuous demand function.

(b) Discrete demands.

Figura A.6: Períodos de tiempos iguales (Fuente: Multi-level lot-sizing and scheduling [19])

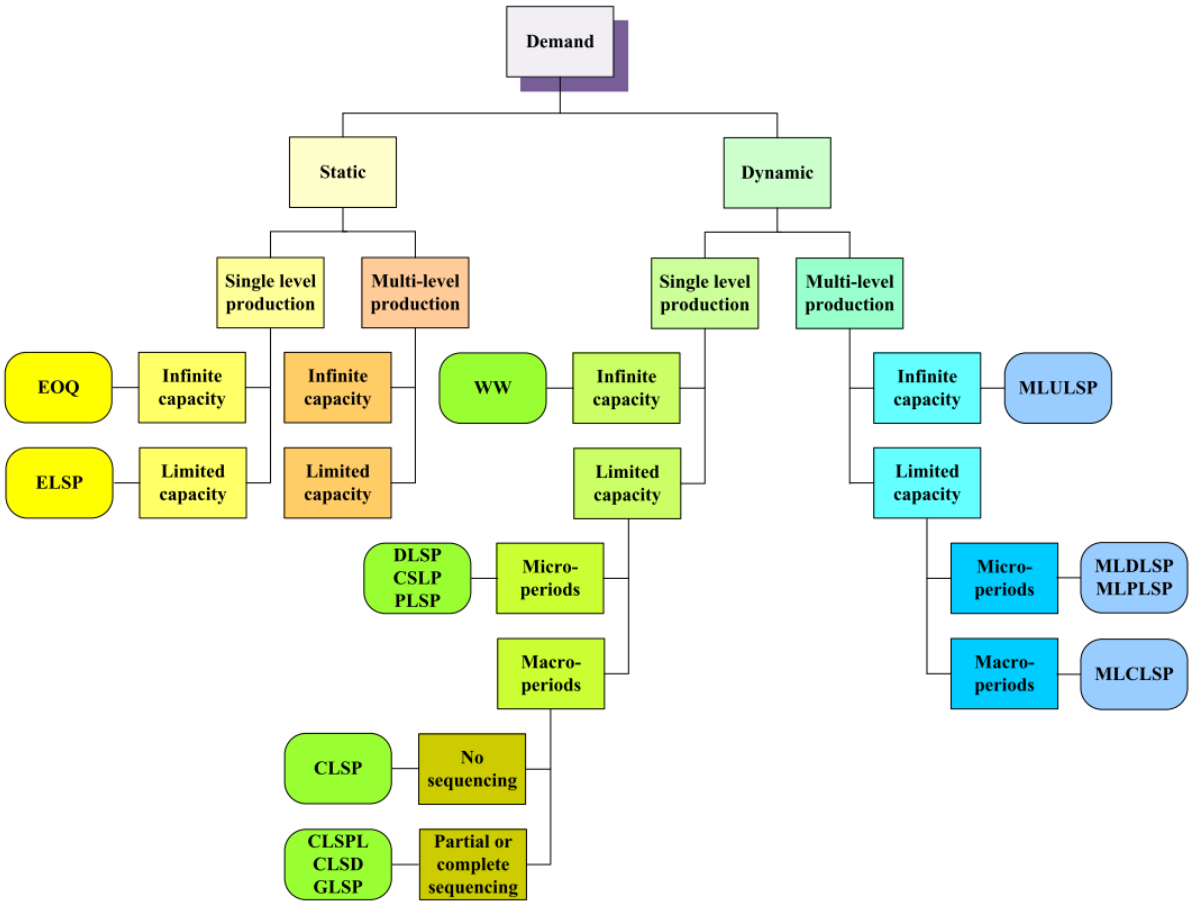


Figura A.7: Taxonomía de lot-sizing (Fuente: tesis [20, p. 57])

Formulación del DLSP

Discrete Lot-Sizing and Scheduling Problem tiene como objetivo determinar el ordenamiento con menor costo de producción para la fabricación de muchos artículos utilizando solo una máquina. Fleischmann (1990)[21] propuso un modelo genérico para DSLP de carácter single-resource, multi-item, single-level, small-bucket y sequence independient cost. La principal diferencia entre los otros modelos es la política de producción: en un periodo se produce a capacidad máxima o no se produce nada ("all-or-nothing"). A continuación se plantea su formulación matemática:

Indices:

$j = 1...M$ productos

$t = 1...N$ periodos

Datos:

d_{jt} : demanda del producto j en el periodo t

p_j : tasa de producción para el producto j (unidades por periodo)

k_t : tiempo disponible en el periodo t (proporcional al largo del periodo)

h_j : costo de mantenimiento de inventario del producto j (por unidad y por periodo)

S_j : setup-cost del producto j

I_{j0} : stock inicial del producto j

s_{jt} : stock de seguridad del producto j al final del periodo t

Variables de decisión:

x_{jt} : cantidad del producto j a ser producido en el periodo t

I_{jt} : inventario del producto j al final del periodo t

Formulación:

$$\min \sum_{j,t} S_j \; max(0, y_{jt} - y_{j,t-1}) + h_j I_{jt} \tag{A.1}$$

$$\tag{A.2}$$

sujeto a:

$$I_{jt} = I_{j,t-1} + p_j y_{jt} - d_{jt} \; \forall j, t \tag{A.3}$$

$$I_{jt} \geq s_{jt}, y_{jt} \in \{0, 1\} \; \forall j, t \tag{A.4}$$

$$\sum_{j} y_{jt} \leq 1 \text{ para todo } t \tag{A.5}$$

$$\delta(x) = 1 \text{ si } x > 0, \text{sino } 0 \tag{A.6}$$

$$y_{jt} = \delta(x_{jt}) \tag{A.7}$$

Revisión de literatura

El modelo original de DLSP ha sido extendido desde single-item a multi-item y single-resource a multi-resource. En adición se han aplicado nuevas restricciones y estructuras de costos. Otros estudios propusieron y/o reforzaron la formulación del modelo para resolver problemas grandes y con instancias más complejas [22]. En consecuencia muchas técnicas de resolución han sido propuestas.

La técnica Programación Dinámica (Dinamic Programming) ha sido desarrollada en muchos casos especiales de DSLP, su funcionamiento se basa en la estructura del problema para encontrar la solución óptima. Su principal desventaja es que ha sido aplicada a escenarios single-item y el pasaje al multi-item no es directo. A pesar de esto, este tipo de técnicas han jugado un rol importante debido a que han sido core de otros algoritmos eficientes.

Por otro lado técnicas como Relajacion Lagrangiana (LR) y Descomposición Dantzig-Wolfe (DWD) han sido aplicados en varias investigaciones; la idea básica consiste en dividir el problema en sub-problemas más sencillos de resolver, LR y DWD resuelven estos sub-problemas. Desde el punto de vista práctico estas dos técnicas tienen

problemas de convergencia; LR ofrece un lower-bound en pocas iteraciones pero la calidad es baja mientras que DWD ofrece lower-bound y upper-bound pero a un ritmo más lento. Otra desventaja es que deben ser extendidas por heurísticas o complementadas con Branch-and-Bound (BB) para obtener buenas soluciones. En escenarios complejos, algunas formulaciones conllevan el uso de muchas variables y a nivel computacional demanda mucho uso de CPU y memoria; para lidiar con este problema LR y DWD han sido combinadas con la técnica Column Generation (CG). CG se compone de dos problemas: master-problem y sub-problem; el master-problem es el problema original considerando algunas variables; sub-problem es el encargado de determinar que variable se debe agregar al master-problem. Este proceso se realiza iterativamente para determinar las variables que hagan aporten a la solución y descartando las que no, en consecuencia el menor uso de variables agiliza el proceso de análisis y búsqueda utilizando información relevante. Las técnicas nombradas forman parte de las clases Mathematical Programming Heuristics y Lagrangean Heuristics [23]; como heurísticas ayudan a obtener una buena solución inicial mejorando tempranamente el proceso de poda de Branch and Bound. De las investigaciones realizadas se encuentra:

The DLS and scheduling problem (1990)

Autores: Fleishcmann

Descripción: En este estudio se busca resolver DSL de caracter single-resource, multi-item y con objetivo minimizar costos. Como punto de referencia se compara con el modelo Capacitated Lot-Sizing Problem (CLSP) en términos de costos y uso de recursos de cómputos.

La estrategia de resolución se basa en la combinación de Branch & Bound (BB) y Lagrangean Relaxation (LR). Debido a que BB utiliza una cota inferior y superior respecto al nodo actual para podar el árbol, LR tiene el rol de establecer la cota inferior de la solución actual; la cota superior es el costo de la última mejor solución factible. El proceso de selección producto-periodo consiste en ir fijando un producto desde el último periodo hasta el primero en función al criterio determinado por LR. Las pruebas de varias instancias dieron como resultado: 1)Los costos de holding y setup obtenidos por parte de DLSP fueron menores a los de CLSP, 2) computacionalmente las instancias mayores a 96 periodos DSLP se resolvieron más rápido que CLSP, el motivo principal se debe a la discretización de DSLP sobre CLSP, 3) por último la aplicación de LR resulta ser una buena forma de extender BB para determinar las cotas inferiores[21].

A Dual Ascent and Column Generation Heuristic for the DLS with Setup Times (1993)

Autores: Cattrysse, Dirk Salomon, Marc Kuik, Roelof Wassenhove, Luk N Van

Descripción: Al igual que el artículo anterior, se pretende resolver DLSP considerando: setups-time, single-resource, producción multi-item y con objetivo la optimización de costos. Para esto se propone implementación que combina las heurísticas Dual Ascent (DA) y Column Generation (CG), donde se pretende buscar una solución óptima o aproximada.

CG es una variante de BB y en vez de considerar todas las variables de una vez, CG vá agregando progresivamente las que más aportan a la solución. La principal ventaja de CG es que puede escalar fácilmente en problemas grandes. GC se divide en dos problemas: master-problem y sub-problem. Aquí el sub-problem es resuelto utilizando Dynamic Programming (DP) y se encarga de obtener una solución al problema Single Item Capacitated Lot-Sizing Problem. Por otro lado para estimar una solución factible del master-problem se calcula la cota inferior y superior: la cota inferior se determina a través de la heurística Dual Ascent (su principal ventaja sobre Lagrangean Relaxation es que converge rápidamente), mientras que la cota superior se determina utilizando las variables agregadas por CG.

Como resultado se obtuvo que: 1) de las 420 pruebas realizadas el 82 % son óptimas y el resto difieren de la óptima en un 1.5 % en promedio, 2) desde el punto de vista computacional para instancias chicas y medianas el tiempo de procesamiento fue de 5 minutos aprox., no obstante la heurística de Fleischmann obtiene resultados más rápido, sin embargo la calidad de solución del algoritmo propuesto acá es mejor. Esto se debe a que Fleischmann saca provecho de la estructura del problema al no considerar setup-time [24].

A linear description of the DLS (1994)

Autores: Stan and Kolen

Descripción: En este artículo se presenta una nueva formulación de DSLP partiendo de un escenario single-item, single-resource y sequence independient setup. La nueva formulación se obtiene al dividir variables (spliting variables). La principal idea de esta técnica es reducir la cantidad de restricciones pero aumenta la cantidad de variables.

En base a la formulación original:

Parámetros:

d_t: demanda del item en el periodo t

f_t: start-up cost del item en el periodo t

p_t: costo unitario de producción del item en el periodo t

h_t: costo unitario de guardar el item en inventario en el periodo t

Variables:

x_t: producción del item en el periodo t

y_t: 1 si el start-up es aplicado en el periodo t; 0 en otro caso

I_t: nivel del inventario al final del periodo t

$$\text{min} \quad \sum_{t=1}^{T}(f_t y_t + p_t x_t + h_t I_t) \tag{A.1}$$

$$\text{s.a} \quad x_t + I_{t-1} = d_t + I_t,\ 1 \leq t \leq T \tag{A.2}$$

$$y_t \leq x_t \leq x_{t-1} + y_t \leq 1,\ 1 \leq t \leq T \tag{A.3}$$

$$I_t \geq 0, 1 \leq t \leq T$$

$$x_t, y_t \in \{0,1\},\ 1 \leq t \leq T$$

La eliminación de la restricción (2) produce que el número de restricciones y variables sea reduzca. Por otro lado los autores proponen el spliting de la variable demanda por periodo en el cual la producción toma lugar.

A linear description of the DLS (Continuación) (1994)

Autores: Stan and Kolen

Descripción: Finalmente el problema queda formulado como:

$$\min \quad \sum_{i=1}^{D}\sum_{t=1}^{t_i}(f_t y_{t,i} + c_t x_{t,i}) \tag{A.1}$$

$$\text{s.a} \quad \sum_{t=1}^{t_i} x_{t,i} = 1,\ 1 \le i \le D \tag{A.2}$$

$$y_{t,i} \le x_{t,i} \le x_{t-1,i-1} + y_{t,i} \le 1,\ 1 \le i \le D,\ i \le t \le t_i t \tag{A.3}$$

$$\sum_{i:t_i \ge t} x_{t,i} \le 1,\ 1 \le t \le T \tag{A.4}$$

$$\sum_{a=t}^{t_i} x_{a,i} \le \sum_{a=i+1}^{t_{i+1}} x_{a,i+1},\ 1 \le i \le D;\ i \le t \le t_i \tag{A.5}$$

$$x_{t,i}, y_{t,i} \in \{0,1\},\ 1 \le i \le D,\ i \le t \le t_i$$

Siendo:

$$x_{t,i} = \begin{cases} 1 \text{ si hay producción en el periodo t para la demanda} \\ \text{en el periodo } t_i \\ 0 \text{ en otro caso} \end{cases}$$

$$y_{t,i} = \begin{cases} 1 \text{ si en el periodo t se inicia un batch cuya primera unidad} \\ \text{de producción se utiliza para satisfacer la demanda del} \\ \text{período } t_i \\ 0 \text{ en otro caso} \end{cases}$$

Como resultado de la re-formulación de DSLP se obtuvo que: 1) como principal ventaja la producción se puede relacionar con la demanda más efectivamente, 2) el orden del problema es $O(D^N T)$, siendo D la demanda, N la cantidad de items y T la cantidad de periodos, para el tipo de escenario analizado no es práctico y no escala para instancias grandes por ser de orden exponencial[25].

En el ámbito de meta-heurísticas, su aplicación ha tenido un rol importante dao que han ofrecido buenos resultados en problemas más complejos y amplios de carácter multi-level, multi-resource y sequence-dependente en comparación con los métodos exactos aplicados[26, p. 22]. Por ejemplo el estudio realizado en [27] aplica Tabú Search para resolver DLSP considerando el escenario multi-item, multi-resource, costos de producción, setup, inventario y backlogging. En su trabajo plantean la formulación MIP (Mixed Integer Linear Programming) de la variante de DSLP y la descripción de la meta-heurística. Como resultado obtuvieron que para instancias chicas del problema la implementación MIP genera una mejor solución en términos de calidad; en instancias grandes Tabu Search obtiene la mejor solución en una hora mientras que MIP lo hace un 5 horas.

A pesar de que no hay muchas meta-heurísticas aplicadas a DLSP, el modelo Capacitated Lot Sizing Problem (CLSP) se puede considerar como punto de partida. CLSP es de índole large-bucket y varios tipos de artefactos pueden ser producidos en la misma máquina y mismo periodo. El modelo genérico es similar a DSLP y la diferencia principal es su política de capacidad y setup [26]. CLSP es una de las sub-clases más estudiadas de Lot-Sizing, algunas publicaciones interesantes son (por más referencias en el artículo [23] hay una completa revisión de la literatura sobre CLSP):

Evolutionary Algorithms for Production Planning Problems with Setup Decisions (1999)

Autores: Hung

Descripción: El objetivo de este artículo es resolver el problema CLSP en un escenario con multi-item, multi-resources, multi-period, setup-time y setup-cost. Estas características reflejan más problemas realistas en los procesos de fabricación. Para resolver este problema se propone el uso de Algoritmos Genético (GA). La novedad de la solución es la presentación de un nuevo operador de reproducción denominado "Sibling"que lidia con los problemas del clásico GA.

Un Sibling se obtiene cambiando el valor de un gen particular de un cromosoma; el nuevo operador es utilizado en un método de ranking para ordenar los cromosomas obtenidos al aplicar sibling, en adición la probabilidad de la mejor ubicación del sibling se obtiene a través de una técnica propuesta por Chen[28]. Este operador se puede ver como un neighborhood move, extendiendo así la búsqueda de GA al generar nuevas generaciones.

Se obtuvo como resultado que al aplicar solo este operador y removiendo los de mutación y cruzamiento se performa mejor la búsqueda en GA, dado que usar el op. de cruzamiento es análogo a realizar una búsqueda global, la mutación es una búsqueda de soluciones vecinas aleatoria mientras que sibling es una búsqueda analíticamente orientada [29].

CLSP by using multi-chromosome crossover strategy (2008)

Autores: Süer, Gürsel A. Badurdeen, Fazleena Dissanayake, Nishantha

Descripción: En este artículo se enfoca en resolver CLSP considerando las siguientes características: single-level, single-resource, sequence-independent setup-time y costos. Para esto se lleva a cabo el uso de la meta-heurística GA haciendo enfoque en el desarrollo de la representación de los cromosomas y en la propuesta de un nuevo operador de cruzamiento.

Cada cromosoma tiene $n = r \times m$ genes, siendo m la cantidad de productos a producir y r el número de periodos de planificación. Los primeros r genes representa el orden del primer producto, los $r + 1$ a $2r$ representa el orden del segundo producto, etc.

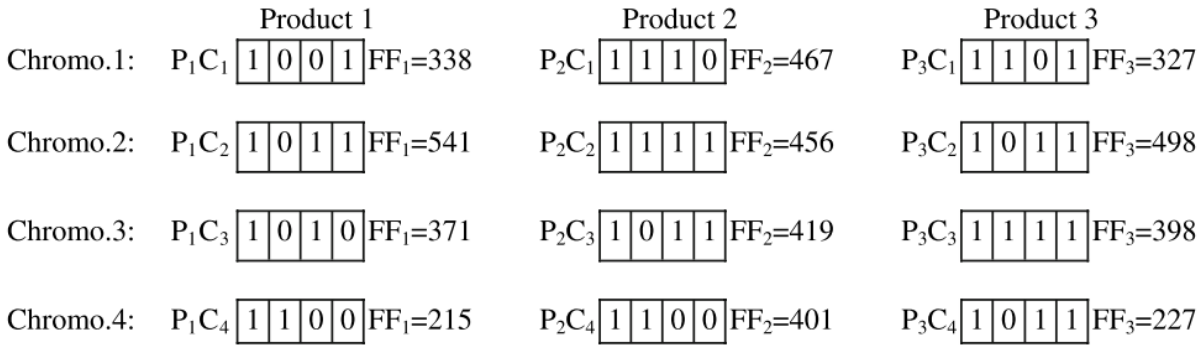
Por otro lado el operador de cruzamiento utiliza múltiples cromosomas en vez de dos comparado con el clásico GA. El procedimiento comienza asignando un fitness a cada producto de cada cromosoma, posteriormente se ordenan los productos con menor fitness considerando todos los cromosomas (ver Figura A.8). Del nuevo conjunto de cromosomas un porcentaje $X\,\%$ se asigna al Grupo 1 (G1), $Y\,\%$ al Grupo2 (G2) y $Z\,\%$ al Grupo3 (G3). Luego por cada producto se realizan los siguientes pasos:

- Se selecciona un producto aleatoriamente de G3 y G1, se cruzan y el mejor producto obtenido se reemplaza en G3 por el que se había seleccionado.
- El mismo proceso se realiza seleccionando un producto de G1 y G2
- Por último se repite para dos productos de G1.

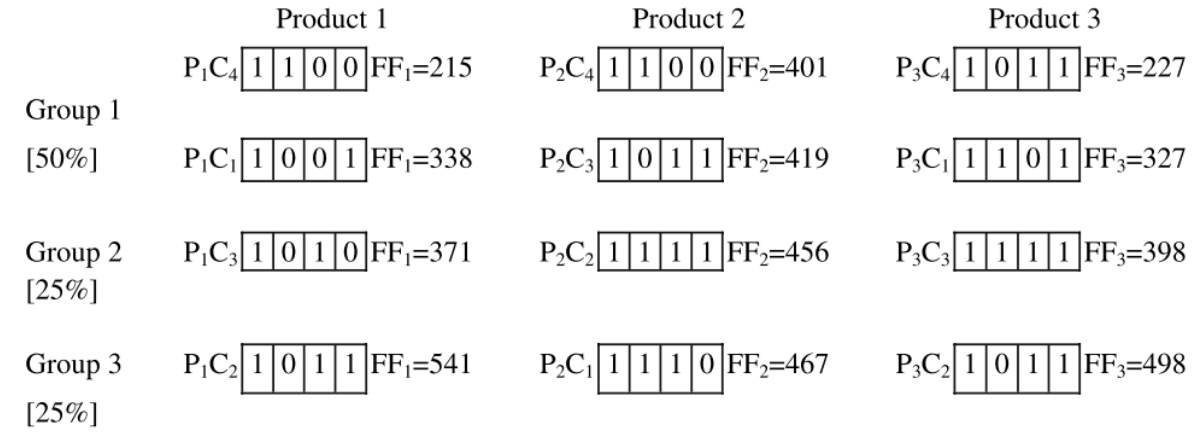
El principal objetivo de esta forma de cruzamiento es identificar buenos y malos segmentos, realizando un cruzamiento selectivo. Es importante ver que la aleatoriedad (diversidad en términos de GA) no se pierde. Comparado con el operador Sibling presentado en el artículo anterior, este requiere un tiempo de procesamiento más largo, sin embargo por ser independiente por producto se puede realizar de forma paralela; sin embargo la ventaja de este operador es que analiza más posibilidades de cruzamiento.

Como resultado se obtuvo: (1) el procesamiento multi-cromosoma no afecta mucho en problemas de gran escala, (2) ante el análisis de varias estrategias que consideran porcentajes de tamaño en los grupos y probabilidades de mutación y cruzamiento se obtiene que el operador aumenta la frecuencia de soluciones óptimas y rendimiento respecto al clásico GA.

A modo de ejemplo se analizarán a continuación aplicaciones de DLSP en el ámbito industrial.



(a) Conjunto de cromosomas y fitness de cada producto



(b) Productos ordenados según fitness

Figura A.8: Ejemplo de operador de cruzamiento[30]

Caso de estudio: DSLP en la industria de neumáticos

En el proceso de fabricación de neumáticos la vulcanización es la última etapa de la cadena, suele ser donde más se genera cuello de botella debido al tiempo de procesado requerido, por ser centralizado y por el alto consumo de energía [31]. Por lo general este proceso consume del 60 % al 90 % de los recursos de fábrica lo que se requiere que sea rápido, confiable y de buen rendimiento [5]. En la literatura revisada el ordenamiento de la producción se hace en función a la demanda resultando un sistema de producción make-to-order. Para esto se suele generar un inventario de ruedas verdes para satisfacer la demanda en la marcha. Una rueda verde se forma por capas y componentes pero le falta la etapa de vulcanizado para lograr lo consistencia final.

El vulcanizado requiere de tres parámetros básicos: una especificación, un molde y un heater (calentador). Una especificación indica entre otros el tiempo de cocción, la presión y la temperatura para lograr la consistencia requerida de la rueda. Un molde puede contener solo una rueda verde, forma la ranuras de la rueda y son de tamaños variables, por ende no se pueden usar en cualquier heater. El heater es la maquinaria utilizada para formar la rueda; estos pueden variar en el tamaño del molde que son capaz de albergar, la cantidad de moldes que puede procesar al mismo tiempo, el consumo de energía y rendimiento. Al momento de estar en producción el cambio a un nuevo molde requiere una pre configuración en el heater, entre ellas limpieza y pre-calentado, por lo que el cambio frecuente genera grandes perdidas de tiempo, energía y oportunidad de capital. Para evitar estos problemas se produce un mismo tipo de rueda en tandas prolongadas, jornadas laborales o días.

En esta industria el modelo DSLP se ajusta de buena forma dado que este problema se puede modelar como un problema DSLP del tipo multi-resource, multi-item, single-level y small-bucket, donde ademas se agregan los setup-cost. A continuación se nombran algunos estudios de la literatura donde se aplica DSLP en la industria de fabricación de neumáticos.

Jans, Raf and Degraeve, Zeger [2] En esta publicación se propone un modelo y algoritmo que resuelve el problema de planificación de producción en la fábrica internacional de neumáticos Solideal. El panorama al que se enfrentan es similar al que se describió anteriormente:

- La demanda está basada en la forma make-to-order.
- Por ser DSLP se considera la política all-or-nothing.
- Se consideran tres costos: mantener inventario, costo de start-up y backlogging.
- Sólo se cuenta con dos tipos de heaters con distinta eficiencia, además muchos tipos de neumáticos se pueden producir en un heater y algunos en ambos.

El algoritmo se basa en el proceso de Column Generation (CG) combinado con Lagrange relaxation (LR) y Branch and Bound sobre las columnas generadas. El hecho de utilizar dos tipos de heaters tiene como ventaja el desarrollo eficiente del algoritmo, sin embargo a nivel práctico es un caso muy acotado y no abarca un panorama más general. Respecto a los resultados obtenidos, aplicar LR sobre CG ayuda a obtener soluciones de forma rápida y sin perder calidad.

Yu, Shengping and Yang, [3] En este artículo se propone un algoritmo que soluciona el problema de vulcanizado aplicando dos etapas, el objetivo es reducir costos y makespan. Se pretende resolver el problema de una compañía contando con 216 máquinas de vulcanizado y una producción diaria de 10,000 neumáticos; asumiendo batches de 50 neumáticos, se va a contar con más de 200 batches que deben ser asignados a las 216 máquinas. Las etapas definidas son distribución de carga y job-scheduling; la distribución de carga consiste en disponer los equipos de procesamiento en cada batch de neumáticos; job-scheduling obtiene el orden de procesamiento en las máquinas asignadas a cada batch.

A pesar de que el modelo no es del tipo DSLP se pueden aplicar algunas de sus restricciones que se adecuan a este caso. Por ejemplo la distribución de carga en las máquinas es una forma de reducir costos y entre otros ajustes como quitar los batches y aplicar la política "all-or-nothing" reduce los setup-cost entre batches de distinto tipos de neumáticos.

En las pruebas realizadas se concluye que el algoritmo de dos etapas es rápido (tiempo de procesado 60 minutos en promedio), reduce costos y aumenta la productividad.

Job Shop Scheduling (JSP)

Introducción

El ordenamiento (scheduling) ha sido uno de los problemas más importantes en cuanto a aplicaciones del mundo real en los último años, en particular, desde los años 50 JSP ha ganado importancia debido a su demanda en la industria y su complejidad. JSP es un problema complejo de optimización combinatoria donde Garey y Ullman [32] demostraron que pertenece en la clase NP-Hard.

Desde que la instancia 10x10 (10 máquinas, 10 tareas) presentada por Fisher y Thompson, JSP ha sido tratado con técnicas convencionales y métodos exactos, tales como Relajacion Lagrangiana, Branch and Bound, Shifting Bottleneck. Los métodos exactos pueden garantizar óptimos globales, sin embargo el tiempo computacional puede incrementar a medida que la dimensión del problema crece. Posteriormente diferentes metodologías inspiradas en procesos naturales, biológicos y físicos han sido propuestos. A pesar de que estas técnicas aplican a varios problemas de optimización, aplican en especial a JSP. Algunas de ellos son Genetic Algorithm (GA), Ant Colony Optimization (ACO), Tabu Search (TS), Simulated Annealing (SA) y Particle Swarm Optimization (PSO) [32].

A todo esto y por ser JSP un modelo genérico se han propuesto extensiones que permiten modelar mejor los problemas del mundo real; Flexible Job-Shop Scheduling y Job-Shop Scheduling with Setup Time son algunos de ellos que se tratarán en las siguientes sesiones.

A continuación se plantea la formulación de JSP y posteriormente te tratará más en detalle los avances en resolución y aplicaciones.

Formulación

El clásico JSP corresponde a un tipo de problemas de planificación de tareas, donde unos de sus principales objetivos es reducir el tiempo total de realización (makespan). Más precisamente se puede describir como un conjunto de m máquinas y n tareas, cada tarea se compone de una secuencia ordenada de operaciones, el cual se precisa que sea procesada de forma continua durante un periodo de tiempo en una máquina dada. Se dá como asumido que cada operación perteneciente a una tarea utiliza máquinas distintas. Además se impone al problema un conjunto de restricciones: 1) una operación que está utilizando una máquina no puede ser interrumpida; 2) no existe restricciones de presidencia respecto a las operaciones de distintas tareas; 3) una máquina puede ser utilizada solamente por una operación a la vez. En [33] se propone tres clases para clasificar los problemas JSP:

Tiempo de procesado Si el tiempo de procesado por cada operación es conocido case dentro de la clase DJSP (Deterministic Job Shop Scheduling Problem); en caso contrario se clasifica como UJSP (Uncertain Job Shop Scheduling Problem).

Cantidad de funciones a optimizar Si la instancia de JSP optimiza sólo una función tal como makespan o tiempo de espera se considera como Single-Object JSP, por el contrario se considera Multi-Object JSP.

Soporte de máquinas Si una operación puede ser procesada en un subconjunto de máquinas se considera esta instancia de JSP como Flexible JSP; en caso contrario, si una operación puede ser procesada sólo en una máquina, esta es conocida.

Varios autores han propuesto formulaciones matemáticas a lo largo de los años; para este relevamiento, se usará como referencia la propuesta por Miloš Šeda [34].

Se asume tres conjuntos finitos: J representa al conjunto de tareas $1, .., n$, M las máquinas $1, ..., m$ y O al de las operaciones $1, .., N$. Se denota como J_i la tarea en que la operación i pertenece, M_i la máquina que será usada por la operación i, t_i el tiempo de comienzo de la operación i y p_i es el tiempo de procesamiento requerido por la operación i.

En el conjunto de operaciones O se define la operación binaria $\rightarrow$ representando la restricción de precedencia entre operaciones de una misma tarea. Por lo tanto, si $i \rightarrow j$ entonces $J_i = J_j$ y no existe $k \in \{i, j\}$ tal que $i \rightarrow k \wedge k \rightarrow j$. Además, por la especificación del JSP, si $i \rightarrow j$ entonces $M_i \neq M_j$.

El problema de optimización de JSP consiste en encontrar un t_i por cada operación $i \in O$ tal que :

$$\begin{aligned}
&\min && max\{t_i + p_i\}, \forall i \in O && \text{(A.1)}\\
&\text{s.a} && \forall i \in O : t_i \geq 0 && \text{(A.2)}\\
& && \forall i, j \in O, i \rightarrow j : t_j \geq t_i + p_i && \text{(A.3)}\\
& && \forall i, j \in O, i \neq j, M_i = M_j : (t_j \geq t_i + p_i) \vee (t_j \geq t_i + p_j) && \text{(A.4)}
\end{aligned}$$

La restricción (3) representa las restricciones de precedencia y la (4) significa que una máquina puede procesar una operación a la vez. Dado que las ecuaciones planteadas no sirven para obtener un scheduling, hay que traducirlas a un modelos de Programación Lineal, para que las funciones objetivos y restricciones sean lineales.

Sea n_i la cantidad de operaciones para de la tarea J_i y $f(i)$ el total de operaciones incluidas en las tareas $\{1, .., i\}$. Se tiene entonces que: $f(0) = 0$, $f(j) = \sum_{i=1}^{j} n_i$, $N = \sum_{i=1}^{n} n_i$.

Utilizando la cantidad de operaciones en las primeras i tareas, se asigna a las n_j operaciones de la tarea j el rango de números entre $[f(j-1)+1, f(j-1)+n_j]$, donde $f(j-1)+n_j = f(j)$. Por lo tanto, la condición (3) se puede expresar como:

$$\forall j \in J, i \in [f(j-1)+1, f(j)-1] : t_{i+1} \geq t_i + p_i$$

Para traducir la ecuación (4) se utiliza la variable binaria $x_{ij} \in \{0, 1\}$ definida como:

$$\forall i, j \in O, i \neq j, M_i = M_j$$
$$x_{ij} = \begin{cases} 1, t_j \geq t_i + p_i, \ (i \rightarrow j) \\ 0, t_i \geq t_j + p_j, \ (j \rightarrow i) \end{cases}$$

Sea T una cota superior del makespan, entonces la ecuación (4) queda como:

$$\forall i, j \in O, i \neq j, M_i = M_j : \begin{cases} t_j \geq t_i + p_i x_{ij} - T x_{ij} \\ t_i \geq t_j + p_j(1 - x_{ij}) - T x_{ij} \end{cases}$$

Por último, sea C_{max} el makespan, se cumple que $\forall j \in J : C_{max} \geq t_{N_j} + p_{N_j}$, por lo tanto el problema queda formulado como:

$$\begin{aligned}
&\min && C_{max} && \text{(A.1)}\\
&\text{s.a} && \forall i \in O : t_i \geq 0 && \text{(A.2)}\\
& && \forall j \in J, f(j-1)+1 \leq i \leq f(j)-1 : t_{i+1} \geq t_i + p_i && \text{(A.3)}\\
& && \forall j \in J : C_{max} \geq t_{f(j)} + p_{f(j)} && \text{(A.4)}\\
& && \forall i, j \in O, i \neq j, M_i = M_j : \begin{cases} x_{ij} \in \{0, 1\} \\ t_j \geq t_i + p_i x_{ij} - T x_{ij} \\ t_i \geq t_j + p_j(1 - x_{ij}) - T x_{ij} \end{cases} && \text{(A.5)}
\end{aligned}$$

Técnicas de resolución

Para los problemas de JSP se encuentran distintos enfoques de resolución, según [35] las técnicas de ordenamiento pueden ser divididas en dos grupos: Tradicionales y No Tradicionales (ver Figura A.9).

Tradicionales Como se menciona en el Anexo de Algoritmos, las técnicas tradicionales se conocen como métodos exactos. En caso de existir solución, este enfoque asegura la obtención de un óptimo global. Algunas de las técnicas utilizadas han sido Dynamic Programming, Mathematical Programming, Column Generation Method, Integer programming, Branch- and-Bound y Mixed Integer Linear Programming.

Ventajas

- Los problemas de ordenamiento se pueden manejar fácilmente con técnicas tradicionales.
- Proporciona alta flexibilidad en el volumen de producción al tener pequeños incrementos en capacidad de producción.
- Genera pocas pérdidas.

Desventajas

- Su uso en escenarios complejos necesitan mucho tiempo computacional.
- No son capaces de lidiar con problemas multi-objetivo (optimizar varias funciones).
- La mayoría de los problemas de scheduling pertenecen a la clase NP-Hard, esto disminuye el rendimiento de las técnicas tradicionales dificultando su diseño.
- En particular, resolver Branch and Bound requiere mucho recursos de cómputos.

Debido a estas limitaciones la tendencia a técnicas no-tradicionales como forma de resolver JSP ha aumentado.

No Tradicionales Las No Tradicionales son conocidas más por métodos de aproximación, son técnicas robustas y no aseguran una solución óptima. Varios de los algoritmos están inspirados en fenómenos físicos, biológicos y naturales, motivo por el cual han recibido críticas por el poco conocimiento matemático requerido. Los más frecuentes en su uso son Local Search Techniques (Ants Colony Optimization, Iterative Methods, Genetic Algorithm, Tabu Search, Simulated Annealing), Particle Swarm Optimization, GRASP y Artificial Neural Network.

Ventajas

- Proporcionan óptimos globales para problemas de ordenamiento.
- El espacio de búsqueda se amplía más.
- Exploran un conjunto más amplio de combinaciones brindando así información extra para futuras generaciones.

Desventajas

- Las soluciones son aproximadas, no siempre exactas.
- Sus aplicaciones pueden ser especificas a un problema en particular, dificultando su análisis respecto a otras implementaciones.
- El comportamiento que se obtiene depende en la forma que se resuelve la exploración y explotación del espacio de soluciones.

Respecto a las heurísticas, ante tanta variedad de técnicas, los autores de [33] presentan tres índices basados en la relación entre el encode (representación de una solución del problema en el algoritmo, determina el tamaño del espacio de búsqueda y la calidad de solución) y los fitness value (peso que se le asigna a una solución). Esta relación es un factor importante que determina la convergencia prematura; la relación uno-a-uno entre la estructura de encode y fitness es la mejor forma que se puede encontrar. Los índices definidos son:

Repetition rate of way of encoding(RRWE): Este índice se utiliza para mejorar la forma de encoding.

$$RRWE = \frac{\text{máxima cantidad de repeticiones de fitness value}}{\text{Cantidad de fitness values}} \times 100\,\%$$

Varience of fitness value: VFV es utilizado para evaluar la capacidad de búsqueda del algoritmo que resuelve JSP. Si VFV es muy alto, la forma de encoding no generará buen rendimiento.

$$VFV = \max_{i \in iter} \sqrt{\sum_{t=1}^{popsize} (fitness_{it} - \overline{fitness_i})^2}$$

Donde $fitness_{it}$ indica el fitness value del individuo t en la iteracion i; $\overline{fitness_i}$ es el promedio de todos los fitness values en la iteracion i.

Approximation rate of solutions: ARS se formula como:

$$ARS = \mid \frac{s_{opt} - s^*}{s^*} \mid$$

Donde:

s_{opt} solución óptima obtenida por algorítmo

$s^* = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} O_{ijk}}{m}$

n cantidad de jobs

m cantidad de tareas

O_{ijk} significa que la opeción j-ésima del job i es procesada en la máquina k

Si ARS es cercano a cero, el algoritmo para resolver JSP tiene buen rendimiento respecto a encontrar buenas soluciones.

La relación de estos índices determina la diversidad de la solución y las soluciones óptimas. En el caso de tener un fitness value lejos del promedio y otros están cercas se tiene que VFV es grande pero la diversidad de soluciones es pequeña. RRWE combinado con VFV refleja la diversidad de la soluciones eficientemente. Por otro lado, s^* es una solución ideal para JSP, si s_{opt} es cercano a s^* la heurística implementada es buena respecto a la calidad de solución, además los valores de RRWE y VFV pueden influenciar en el consumo de CPU. La labor de diseñar un encode adecuado puede llegar a ser compleja, sin embargo es posible combinar meta-heurísticas para mejorar el proceso de búsqueda.

En [36] los autores desarrollaron un algoritmo basado en el Bee Colony Algorithm para resolver el clásico problema JSP extendido con una búsqueda local en base a la información obtenida de los jobs con más prioridad. La representación de cada solución utiliza una matriz en forma de arreglo y como resultado se obtuvo soluciones alejadas al fitness value promedio. Proyectando la idea a los índices mencionados se puede ver: 1) la representación utilizada puede genera varios fitness values para distintas permutaciones del arreglo (RRWE alto), (2) al tener valores alejados al del promedio el VFV es alto, por ende el encoding no generará buen rendimiento. A pesar de esto, la ayuda de la búsqueda local permitió obtener soluciones de buena calidad.

En otra parte, los autores de [37] aplican Genetic Algorithm para resolver JSP sin extensiones pero haciendo énfasis en el rol que juega el encoding y decoding de una solución. Su propuesta busca mejorar algunas de las desventajas del clásico GA:

- Limitaciones en la capacidad de búsqueda y convergencia a óptimos locales.
- Para problemas grandes, un gran número de pequeñas operaciones deben ser ejecutadas durante un tiempo prolongado.
- Por ser GA un algoritmo aleatorio y la necesidad de múltiples operaciones se logra que la confiabilidad y estabilidad de las soluciones sea pobre.
- Los operadores de cruzamiento y mutación operan con probabilidad constante.

Para solucionar estas desventajas se propone la heurística Adaptive Genetic Algorithm (AGA); el cual consiste en variar las probabilidades de cruzamiento y mutación en la búsqueda en distintos periodos. En el caso de la mutación, si la probabilidad es baja la generación de nuevos individuos es precoz, de ser muy alta la generación de individuos se comporta como una búsqueda aleatoria; del lado del cruzamiento, una probabilidad alta produce una generación rápida de individuos adicionando el riesgo de perder información de generaciones anteriores, de ser muy lenta se ralentiza la generación de individuos produciendo que el proceso de búsqueda culmine prematuramente. El adecuado uso de las probabilidades garantizan estabilidad en el algoritmos GA. Llevando esta propuesta al enfoque de índices se puede ver que el ajuste de las probabilidades de mutación y cruzamiento ayudan a reducir el

VFV, ya que no se generan fitness values muy aleatorios y enfocados; este comportamiento también mejora el ARS dado que el ajuste permite reducir el rango de aproximación a las soluciones óptimas. Como resultado se obtuvo un algoritmo confiable respecto a calidad de solución e incremento en velocidad de búsqueda y convergencia.

Revisión de Literatura

JSP fue propuesto por primera vez por John F. Muth, y Gerald Luther Thompson [38], desde entonces ha sido un problema clásico de ordenamiento aplicado principalmente a la ingeniería industrial, aunque es utilizado en otras ramas. A continuación se hace una revisión de algunos estudios realizados.

Industrial Scheduling (1963)

Autores: Muth y Thompson

Descripción: Se propone por primera vez JSP [38].

The Shifting Bottleneck (SB) procedure for JSP (1988)

Autores: Egon Balasand Adams

Descripción: Se propone la metodología SB como técnica de resolución de JSP [39].

Genetic Algorithms and JSP (1990)

Autores: John y James

Descripción: Se aplica conceptos de GA a JSP [40].

A Genetic Algorithm for Flowshop sequencing (1995)

Autores: Colin Reeves

Descripción: Se desarrolla una solución para disminuir el makespan para FSP con n trabajos y m máquinas [41].

A Hybrid Genetic Algorithm for JSP (2002)

Autores: José Fernando

Descripción: Se genera un ordenamiento parametrizado y se desarrolla un nuevo procedimiento de búsqueda local [42].

Research on JSP in fuzzy environment (2005)

Autores: Cheng, Li Yan

Descripción: Se desarrolla framework para optimizar JSP con tiempos de procesamiento inciertos.

Clonal Selection Based Memetic Alg. for JSP (2008)

Autores: Jin Hui Yang

Descripción: Se desarrolla una solución que aumenta la inspección y exploración para componer técnicas de búsqueda evolutivas [43].

A Simple Optimised Search Heuristic for the JSP (2008)

Autores: Susana Fernandes, Helena R. Lourenco

Descripción: Se propone una nueva heurística para la búsqueda de óptimos en la resolución de JSP, combinando GRASP y Branch and Bound [44].

A multi-objective PSO for JSP (2010)

Autores: Dy Sha

Descripción: Se presenta una solución para JSP donde se busca optimizar muchas funciones objetivos [45].

An Artificial Bee Colony Algorithm Based on Problem Data Properties for JSP (2011)

Autores: Rui Zhang

Descripción: La implementación presentada extiende la implementación de ABC con una búsqueda local utilizando la información brindada por los jobs más críticos para minimizar el retraso ponderado total [46].

Two enhanced differential evolution algorithms for JSP (2012)

Autores: Wisittipanich, Kachitvichyanukul

Descripción: Se desarrolla algoritmo que optimiza makespan y tardanza haciendo balance de exploración [47].

JSSP with makespan objective: A heuristic approach (2014)

Autores: M. Ziaee

Descripción: Desarrolla nueva heurística que minimiza makespan.

Dual Hybrid Algorithm for JSSP (2014)

Autores: Tuan

Descripción: Desarrolla un algorítmo GA con acceso dual híbrido [48].

A hybrid particle swarm optimization and simulated annealing algorithm for JSP (2014)

Autores: Chuan Ma

Descripción: Combinando SA y PSO se desarrolla que mejora el proceso de búsqueda [49].

Parall el Genetic Algorithm for solving JSP Using Topological sort (2014)

Autores: Somani

Descripción: Se plantea solución de JSP aplicando clasificación topológica para minimizar makespan [50].

An improved algorithm for flexible job shop scheduling (2015)

Autores: Jin-dong, Ying-hong

Descripción: Se desarrolla una implementación de GA que mejora la precocidad, estabilidad y velocidad de búsqueda para resolver JSP [37].

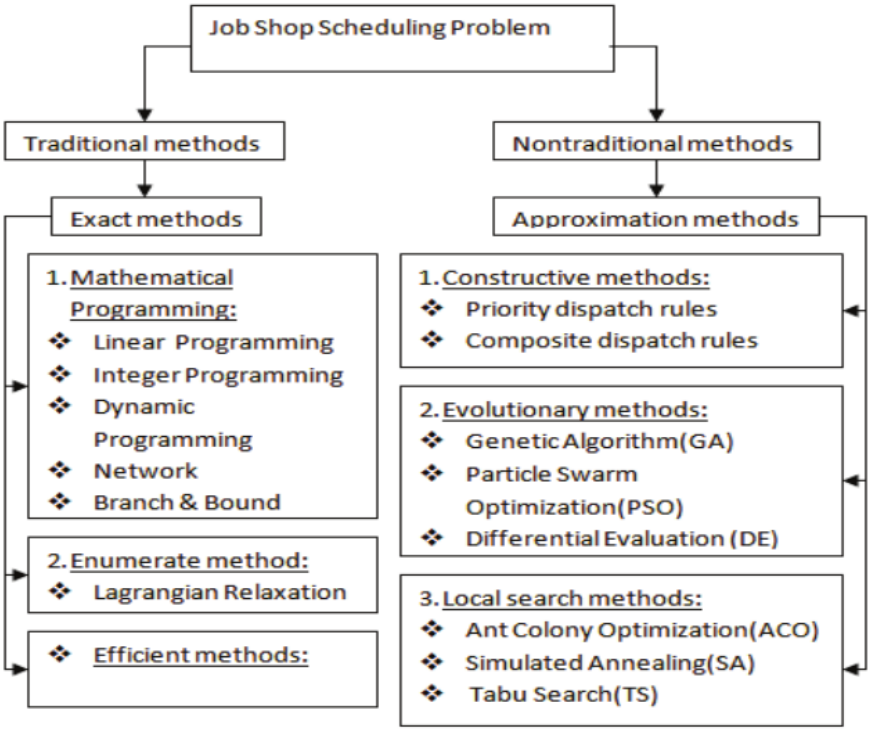


Figura A.9: Clasificación de soluciones para JSP (Fuente: Job Shop Scheduling Algorithms [35])

Extensiones

JSP es muy genérico como para aplicarlo en varios ámbitos específicos; esto ha influenciado en la definición de nuevos modelos que extienden el clásico JSP y permiten estar más próximo a escenarios de la vida real. En las próximas dos secciones se tratarán dos extensiones de JSP: Flexible Job-Shop Scheduling Problem (FJSSP) y Job-Shop Scheduling Problem con Sequence Dependence Setup Time (JSPSDS). Se abarcarán formulación del problema, descripción y estudios recientes sobre su resolución.

JSP with Sequence Dependence Setup Time

Introducción Debido a las dificultades derivadas de JSP muchas simplificaciones han sido hechas; por ejemplo es común asumir que el tiempo de configuración de un job es muy pequeño para considerarse en el problema y en consecuencia es combinado como parte del tiempo de procesado. En situaciones de la vida real estas suposiciones difícilmente aplican [51], el caso se demuestra con la aparición de máquinas multi-propósito como parte de los avances tecnológicos en la maquinaria manufacturera; al momento de utilizarlas en algunos escenarios se debe realizar una serie de configuraciones para que el proceso de producción pueda comenzar.

La magnitud de la tarea consume tiempo y además se pueden hacer en distintas modalidades, en la literatura se encuentran [52]:

Sequence-independent setup times: El tiempo de configuración puede ser considerado como el tiempo de

procesado, o sea, cada configuración es independiente de la tarea ejecutada anteriormente.

Sequence-independent batch setup times: No es necesario configurar las máquinas que realizarán un lote de operaciones del mismo tipo (batchs), la configuración se realiza cuando se pasa de procesar un tipo de batch a otro y no depende del batch procesado anteriormente.

Sequence-dependent batch setup times: Es un caso parecido al anterior y más exigente, donde el tiempo requerido depende del tipo de batch procesado anteriormente.

Sequence-dependent setup times: Este es el caso más general y complejo de todos. Entre cada par de operaciones se requieren tiempos de setup.

JSPSDS siendo una extensión de JSP es igual de complejo en su resolución ya que forma parte de la clase NP-Hard [53], esto justifica también el uso de meta-heurísticas para efectuar su resolución. JSPSDS fué formulado en los años '70, sin embargo este problema ha recibido atención desde el año 2000.

Formulación A continuación se formula el modelo tomando como referencia el planteado en [51];

n = Número de jobs

m = Número de máquinas

J_j = Tarea j, $j = 1...n$

M_i = Máquina i, $i = 1...m$

$O_{i,j}$ = La operación j en M_i

$p_{i,j}$ = Tiempo de procesado de la tarea i en M_i

σ^j = Especificación de sequencia de procesado de las operaciones en la tarea j

s^i_{jk} = Tiempo de setup para procesar la tarea k en la máquina i
Inmediato a la tarea j

C_{max} = Makespan

t_{ij} = Tiempo de inicio para procesar la tarea J_j en M_i

t_j = Tiempo de finalización de la tarea J_j en la máquina M_i

A^i = Conjunto de jobs cuya próxima operación será realizada en M_i

Dada estas definiciones, el problema puede ser formulado como Mixed Integer Programming:

$$\min \quad C_{max} \tag{A.1}$$

$$\text{s.a} \quad t_{\sigma^i_h j} + p_{\sigma^i_h j} \leq t_{\sigma^i_{h+1},j},\ h = 1..m-1,\ j = 1..n \tag{A.2}$$

$$t_{\sigma^i_m j} + p_{\sigma^i_m j} + s^{\sigma^j_m}_{j0} x^{\sigma^j_m}_{j0} \leq C_{max},\ j = 1..n \tag{A.3}$$

$$t_{ij} + p_{ij} + s^i_{jk} x^i_{jk} \leq t_{ik} + (1 - x^i_{jk})M \tag{A.4}$$

$$i = 1,..m,\ j = 1..n,\ k = 1..n,\ k \neq j$$

$$s^i_{0k} x_i 0k \leq t_{ik}, k = 1..n \tag{A.5}$$

$$\sum_{k=0,k\neq j}^{n} x^i_{jk} = 1, i = 1..m,\ j = 0..n \tag{A.6}$$

$$\sum_{j=0,k\neq j}^{n} x^i_{jk} = 1, i = 1..m,\ j = 0..n \tag{A.7}$$

$$x^i_{jk} \in \{0,1\},\ t_{ij} \geq 0$$

donde x^i_{jk} es 1 si el job k es continuo al job j en M_i, sino 0

La restricción (2) asegura la precedencia entre jobs, (3) implica que todos los trabajos deben completarse antes del makespan, (4) asegura que una máquina puede comenzar a procesar sólo si la operación anterior y el setup requerido se completaron, (5) es el caso 31 pero cuando se procesa por primera vez en una máquina. La restricción (6) indica que en una máquina i, cuando un job ha sido completado otro diferente job debe ser seleccionado para procesar después, al menos que la máquina i halla completado todas las operaciones. La restricción (7) indica que una operación de un trabajo debe tener sólo un predecesor, excepto la primera operación de la máquina.

Revisión de Literatura La aplicación de métodos exactos para la resolución de problemas JSP con setup-time no ha sido una buena salida para encarar este problema, por lo que el desarrollo de heurísticas y meta-heurísticas ha sido requerido[54]. A continuación se mencionan algunos avances respecto al tema.

Partiendo que JSPSDS cuenta con un espacio de soluciones de tamaño $(n!)^m$ (n jobs, m máquinas) [51](2001) proponen un algoritmo híbrido combinando Genetic Algorithm (GA) y Heuristic Rules (HR) parar optimizar makespan, dando como resultado: (1) la reducción de $(n!)^m$ a $(n)^m$ del espacio de soluciones (esta reducción puede producir que soluciones óptimas se pierdan, el diseño de la heurística debe ser adecuado y contemplar casos bordes), (2) la introducción de mejoras en el estado inicial de la heurística aumenta el rendimiento durante el proceso de búsqueda. El punto (1) se logra haciendo uso de GA para asignar la primer operación en cada máquina; el (2) utiliza las siguientes reglas: (2.1) los jobs que tienen menos tiempo de procesado más tiempo de setup son los más prioritarios, (2.2) cuando un job es finalizado en una máquina, el siguiente job debe tener asignada una prioridad. El argumento de esta heurística se basa en que la robustez del algoritmo GA genera pérdida de rendimiento durante la exploración del espacio de soluciones, por lo que introducir información extra a través de HR se le puede sacar ventajas. Como desventaja de esta propuesta el encoding de las soluciones para GA están basadas en permutaciones de operaciones, desde el punto de vista de índices no es una buena forma ya que se generan más soluciones con fitness values cercanos y pérdida de tiempo en análisis, sin embargo la ayuda de Heuristic Rules puede evitar este problema. De todos modos los resultados presentados demuestra mejoras en rendimiento y calidad de solución respecto a otros algoritmos de la fecha.

Los autores de [55] propusieron una extensión más sofisticada de GA combinándolo con Local Search: antes de agregar una solución (cromosoma) con un makespan asociado a la población, se explorar el neighborhood aplicando reglas para buscar otra solución con un makespan mejor. En la implementación se analizan tres reglas (N_1^s, N_2^s, N_3^s) que determinan un neighborhood válido respecto a un cromosoma, cada una tiene su complejidad de desarrollo y beneficios. La codificación de los cromosomas es otro aspecto que se consideró importante, un cromosoma es una permutación de un conjunto de operaciones, cada una siendo representada por un número de job. Por ejemplo el cromosoma (2 1 1 3 2 3 1 2 3) representa las operaciones (θ_{21} θ_{11} θ_{11} θ_{32} θ_{22} θ_{32} θ_{13} θ_{23} θ_{33}). Esta forma de representación es fácil de evaluar y permite implementar operadores eficientes; en [56] se comparó con otras doce formas y esta resultó ser la mejor para los problemas de JSP. En las instancias del benchmark utilizado para las pruebas, se obtuvieron mejores soluciones a las encontradas por otros métodos eficientes y con buenos tiempos de computo.

Flexible Job-Shop Scheduling (FJSP)

Introducción JSP se puede plantear como: dado un conjunto $M = \{u_1...u_m\}$ de m máquinas que deben procesar n jobs $J_1, ..., J_n$, donde cada job J_i consiste de n_i operaciones $O_{i,j}$, con $j = 1, ..., n$, en el cual tiene que ser procesadas en el orden $O_{i,1} \rightarrow O_{i,2} \rightarrow ...O_{i,n_i}$.

FJSP es una extensión de JSP donde una operación $O_{i,j}$ puede ser procesada por un conjunto de máquinas $M_{i,j} \in \{u_1, ..., u_m\}$; en contraste con JSP, cada operación puede ser realizada sólo por una máquina [57]. FJSP es más complejo que el clásico JSP y ha sido demostrada su pertenencia en la clase NP-Hard en 1993 [58].

Este modelo cuenta con ciertas modalidades; una de ellas se denomina "Machine flexivity" y refiere a la capacidad de una máquina para realizar diferentes tipos de operaciones. Esta modalidad permite trabajar con pequeños batchs, causando tiempos de entrega más cortos, una mayor utilización de máquinas y un nivel de inventario para producir reducido, dado que se realizan más tareas al mismo tiempo. La modalidad "Routing flexibility" asume la existencia de caminos alternativos a los largo de la cadena de producción. Esta modalidad se relaciona directamente con FJSP y tiene dos etapas básicas que resolver: (1) asignar cada operación a una de las máquinas alternativas, (2) secuenciar el conjunto de operaciones que tiene asignada una máquina. Dada la naturaleza de este modelo puede que sea conveniente optimizar varias funciones objetivos dado que la asignación de operaciones a máquinas puede generar mala carga de trabajo, en especial a las máquina críticas. Routing flexibility se puede extender contando con conjuntos de máquinas idénticas o de multi propósito [59] y esto ha sido uno de los motivos por el cual FJSP tiene una importante relevancia práctica.

Revisión de literatura El primer estudio conocido de FJSP fué propuesto por Brucker and Schlie en 1990 [60] y como ventaja de ser una generalización de JSP permitió heredar varias de las técnicas ya desarrolladas: métodos exactos, heurísticas y metaheurísticas, la Figura A.10 muestra el avance histórico respecto a JSP.

En la literatura revisada a lo largo de este relevamiento, Simulated Annealing (SA) y Tabú Search (TS) han

sido las técnicas más utilizadas, en especial para generar buenas soluciones iniciales. En particular para FJSP, SA ha sido utilizado para secuenciar las operaciones asignadas a máquinas [59, p.5]. TS junto con implementaciones más sofisticadas respecto a la forma de generar neighborhood moves ha permitido construir mejores soluciones iniciales cuando se combina con otras meta-heurísticas, en particular Genetic Algorithm (GA).

Las meta-heurísticas más explotadas en FJSP son GA, Particle Swarm Optimization (PSO) y Ant Colony Optimization (ACO), su combinación con otras han dado como resultado soluciones de buena calidad y con poco consumo de cómputos [60]. A continuación se realiza una breve descripción de los estudios realizados en los últimos años, en la Figura A.11 un gráfico respecto a las publicaciones (la búsqueda fué realizada en el sitio web IEEE Xplore el 15/03/2017 filtrando por título "flexible job shop scheduling ").

GA for the FJSP (2003)

Autores: Kacem, I.

Descripción: Se busca optimizar makespan, carga de trabajo en maquinaria común y critica. La asignación de operaciones a las máquinas se resuelve aplicando "Approach by Location"y la secuencia de las operaciones. en cada máquina se resuelve por GA. Como resultado, en instancias chicas y medianas el algoritmo se comporta de manera aceptable, la representación utilizada en GA no es la más adecuada [61].

Optimization by Phases for the FJSP (2004)

Autores: Zribi, Nozha Kacem, Imed El Kamel, Abdelkader Borne, Pierre

Descripción: Se combinan tres técnicas para resolver FJSP, buscando optimizar makespan y carga de trabajo en máquinas. En la primera fase de asignación de operaciones en máquinas se aplica Tabú Search y Local Search; posteriormente para secuenciar las operaciones en las máquinas se aplica GA. Como forma de mejorar la calidad de solución se realiza una calibración de parámetros en las probabilidades utilizada por los operadores en GA [62].

A novel variable neighborhood Particle Swarm Optimization for multi-objective FJSP (2007)

Autores: Liu, HongboAbraham, AjithGrosan, Crina

Descripción: PSO tiene la desventaja de converger rápidamente, reduciendo drásticamente la posibilidad de obtener buenos scheduling. Los autores enfrentaron tres problemas: (1) mejorar la convergencia incorporando búsqueda local en las partículas, (2) incrementar la escala respecto al tamaño de las instancias de FJSP utilizando PSO y (3) optimizar makespan y la suma de tiempos de realización. Como resultado las soluciones en instancias grandes son mejores respectos a otras implementaciones de PSO [63].

A novel initialization method for solving FJSP (2009)

Autores: Yang, Shi Yang ShiGuohui, Zhang Guohui ZhangLiang, Gao Liang GaoKun, Yuan Kun Yuan

Descripción: En este estudio el problema se enfoca en la creación de la población inicial del algoritmo GA. Este problema es importante dado que la calidad de la población inicial influye en la velocidad de convergencia y en el resultado obtenido. El método se compone de una Selección Global (GS) y una Selección Local (LS). GS se usa para determinar diferentes individuos en la población considerando la carga de máquinas, mientras que LS se utiliza para encontrar una asignación con menor tiempo de ocupación en las distintas máquinas en cada individuo. Estas dos modalidades mejoran la capacidad de exploración. Esta técnica se aplicaron a implementaciones existentes dando como resultado una notoria mejoría en el makespan, reducción de tiempos de cómputos y velocidad de convergencia [64].

A GA with TS for Multi-Objective Scheduling Constrained FlexibleJob Shop (2011)

Autores: Di, LiangZe, Tao

Descripción: Se busca optimizar makespan, costos y utilización de equipamiento en un escenario de FJSP. Para evitar la convergencia prematura del GA clásico se extiende a través del uso de Tabú Search por su capacidad adaptativa. El modelo multi-objetivo se ajusta a varios casos prácticos y puede ser extendido para considerar tiempo ocioso de equipamientos y delays en los jobs. En cuanto a los resultados las simulaciones demostraron ser un algoritmo eficaz y eficiente [65].

FJSP based on multi-population genetic-variable neighborhood search algorithm (2015)

Autores: Liang, X Weiping, Sun Huang, M

Descripción: Con el objetivo de optimizar makespan, carga de máquinas y carga total de máquinas se propone una nueva metodología que busca generar información acumulativa en el proceso de creación de nuevas generaciones en el algoritmo GA. Partiendo desde la población inicial se seleccionan los individuos con mejor fitness almacenándolos en una memoria externa. Posteriormente en la etapa crossover esa información es usada para generar nuevos individuos con los buenos genes ubicados en la memoria externa, de esta manera se logra obtener progresivamente mejores soluciones. La ayuda de una búsqueda local sobre la memoria externa permite buscar una solución óptima. Como resultado se mejora mucho de los aspectos desfavorables de GA [66].

Entre otros estudios se puede encontrar [67] donde los autores desarrollan un algoritmo híbrido entre Tabú Search y una estructura de Neighborhood. En [68] se propone un algoritmo híbrido basado en Particle Swarm Optimization (PSO) y Simulated Annealing. En ([69], [70]) se hace uso de Genetic Algorithm, PSO en ([71], [72]) y por último Ant Colony Optimization (ACO) [73], muchas de estas publicaciones proponen algoritmos híbridos.

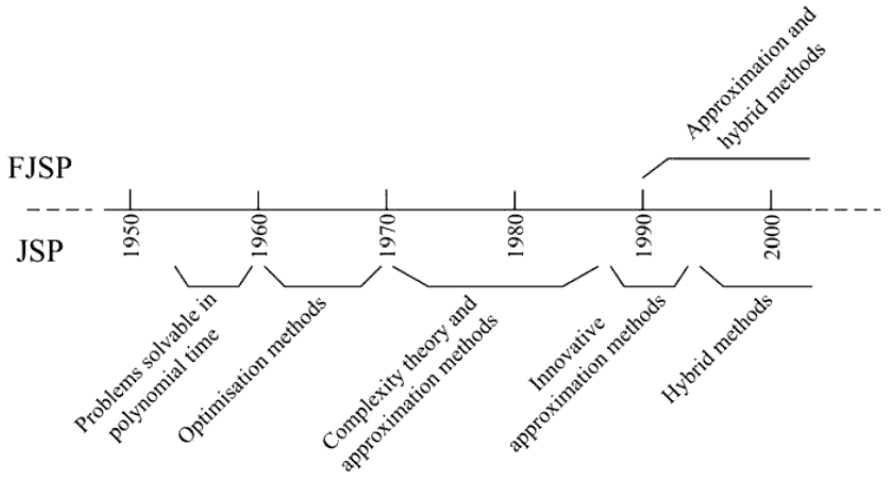


Figura A.10: Avance histórico de JSP y FJSP (Fuente: A Taxonomy for the FJSP [60])

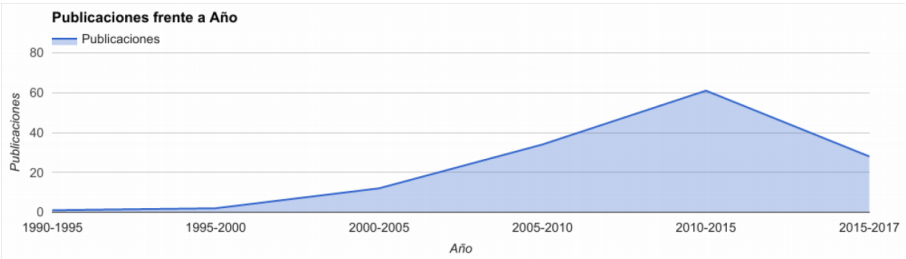


Figura A.11: Publicaciones de FJSP

Conclusiones

El ordenamiento en el ámbito industrial es un campo muy complejo, abarca macro y micro decisiones e influyen en aspectos medio ambientales y económicos. En tiempos donde los plazos de entregas son acotados y la demanda crece se ha visto que su desarrollo a venido acompañado con avances respecto a modelos y algoritmos.

En especial, la pertenencia de los modelos Discrete Lot-Sizing y Job-shop scheduling en la clase NP-Hard a motivado la investigación de muchas heurísticas y meta-heurísticas para encontrar buenas soluciones. Su rol han sido clave y se ha visto como desventaja la calidad de solución no es buena al optar por menos tiempos de análisis y búsqueda, sin embargo la tendencia hoy en día es generar algoritmos híbridos para sacar potencia de cada uno. Como resultado la calidad de solución y tiempo han mejorado notoriamente para instancias de gran escala. Por otro lado las técnicas exactas no son una buena salida para problemas grandes, pero su potencia de encontrar buenas soluciones en instancias chicas ha permitido integrarse en varios algoritmos complejos.

Por último, se debe tener en cuenta que los escenarios industriales han forzado la necesidad de optimizar más de una función objetivo. Por ende la complejidad y necesidad de soluciones aumenta ya que se deben considerar más restricciones y esto dispara nuevos retos.

ANEXO B

Algoritmos, análisis de complejidad y técnicas de resolución

Introducción

¿Los problemas de ordenamiento siempre son complejos? ¿Cómo es posible recorrer el inmenso espacios de soluciones? ¿Es posible siempre obtener la mejor solución? ¿los recursos de cómputos disponibles de hoy en día son capaces de resolver estos problemas?. Estas son algunas de las interrogantes disparadoras de esta sección para estudiar la complejidad en los problemas combinatorios.

Los problemas combinatorios pueden generar espacios de soluciones muy extensos y una mala práctica es revisar cada una de ellas para determinar la mejor. Contando que la mayoría de ellas no son factibles ha sido necesario implementar algoritmos y técnicas que acoten el espacio de búsqueda para así reducir el tiempo de respuesta, si existe solución.

Ejemplo Utilizando como referencia el ejemplo planteado en [16, p. 157] se pretende generar una idea del problema. Suponer que se quiere obtener la tardanza total de un ordenamiento parcial, esto significa: determinar el tiempo requerido por una máquina para realizar n trabajos. Consideramos una CPU lenta si su frecuencia de procesamiento es de 1Ghz, equivalente a 1.000.000.000 operaciones básicas (suma, resta, multiplicación, división) por segundo. Para hacer sencillo los cálculos suponer que el ordenamiento parcial se puede obtener con una sola operación. Con la intensión de generar perspectiva se comparará con una CPU teórica cinco millones más rápida, equivalente a 5.000.000Ghz (actualmente es raro que la frecuencia de una CPU supere los 6Ghz).

Como se mencionó en la sección A existen $n!$ formas de asignar los trabajos en una máquina. En el caso de contar con 12 trabajos el tiempo requerido por una computadora lenta es de $12!/1000000000 = 0,4790016$ segundos; si aumentamos a 15 trabajos, el tiempo necesario es $15!/1000000000 = 21,8$ minutos. Incrementando solo el 25 % del tamaño del problema el tiempo empleado es 2785 veces mayor. En la Tabla B.1 se visualiza mejor la comparativa entre los dos tipos de CPU propuestos.

Existen muchos algoritmos y técnicas para solucionar problemas combinatorios, en la literatura hay dos clases principales: una de ellas se denomina Métodos Exactos, donde garantizan encontrar un óptimo en el tiempo que considere necesario; la otra es conocida por Técnicas de Aproximación, usualmente llamados heurística o meta-heurística, el cual dan una buena solución del problema en un razonable intervalo de tiempo (no garantiza la mejor solución). Ambas técnicas son análogas a una balanza de dos pesas, en una pesa ese encuentra el tiempo de cómputo y en el lado opuesto la calidad de solución.

Los investigadores se han enfrentado al problema de comparar los algoritmos en términos de eficiencia, velocidad y requerimientos de cómputos. El campo de Análisis de Algoritmos ayuda a estimar la cantidad de operaciones requeridas para ejecutar un algoritmo independiente de su implementación. Ofrece ademas una métrica para el análisis de la complejidad. En términos de notación, O representa el orden de un algoritmo; por ejemplo se dice que un algoritmo tiene $O(n)$ si éste requiere de n etapas para culminar su ejecución. En la Tabla B.2 se muestra los órdenes más usados.

n	n!	Tiempo de Computo (lenta)		Tiempo de Computo (rápida)	
1	1	1	ns	0,2	fs
2	2	2	ns	0,4	fs
3	6	6	ns	1,2	fs
4	24	24	ns	4,8	fs
5	120	120	ns	24	fs
6	720	720	ns	144	fs
7	5.040	5,04	us	1,01	ps
8	40.320	40,32	us	8,06	ps
9	362.880	0,36	ms	72,58	ps
10	3.628.800	3,63	ms	726	ps
11	39.916.800	39,92	ms	7,98	ns
12	479.001.600	479	ms	95,8	ns
13	6.227.020.800	6,23	s	1,25	us
14	87.178.291.200	87,18	s	17,44	us
15	1.307.674.368.000	21,79	min	262	us
16	20.922.789.888.000	349	min	4,18	ms
17	355.687.428.096.000	98,8	h	71,14	ms
18	6.402.373.705.728.000	74,1	días	1,28	s
19	121.645.100.408.832.000	3,85	años	24,33	s
20	2.432.902.008.176.640.000	77,09	años	8,11	min
21	51.090.942.171.709.400.000	16,19	siglos	170,3	min
22	1.124.000.727.777.610.000.000	356	siglos	62,44	h
23	25.852.016.738.885.000.000.000	819	milenios	59,84	días
24	620.448.401.733.239.000.000.000	19,66	millones de años	3,93	años
25	15.511.210.043.331.000.000.000.000	492	millones de años	98,3	años
26	403.291.461.126.606.000.000.000.000	12,78	billones de años	25,56	siglos
27	10.888.869.450.418.400.000.000.000.000	345	billones de años	690,09	siglos
28	304.888.344.611.714.000.000.000.000.000	690	años del universo	1,93	millones de años
29	8.841.761.993.739.700.000.000.000.000.000	20,013	años del universo	56,04	millones de años
30	265.252.859.812.191.000.000.000.000.000.000	600,383	años del universo	1,68	billones de años

Tabla B.1: Comparación de tiempos de computo

Otros de los campos que entrar en juego es la Complejidad Computacional: enfocado a clasificar los algoritmos según su estructura y dificultad, encaran más el problema que los algoritmos mismos [16, p. 156]. Hay varias clases importantes que pertenecen a esta rama [74, p.14-15]. Partiendo que un problema de decisión responde sí o no se encuentran las clases:

P: Es una clase de complejidad que representa a todos los problemas de decisión que se pueden resolver en tiempo polinomial. Los algoritmos con orden $O(n)$, $O(logn)$ y $O(n^2)$ son algunos ejemplos.

NP: Representa a todos los problemas en el cual sí una instancia responde a "sí"esta se puede verificar en tiempo polinomial. Esto significa que si alguien da una instancia de un problema donde la respuesta es si, entonces podemos verificar que es correcta en tiempo polinomial. Esta clase incluye todos los problemas de P.

NP-Complete: Representa todos los problemas X en NP en el cual es posible derivar cualquier otro problema NP Y a X en tiempo polinomial. Esto significa que si se sabe como resolver el problema X podemos resolver el problema Y.

Orden	Nombre
$O(1)$	orden constante
$O(logn)$	orden logarítmico
$O(n)$	orden lineal
$O(nlogn)$	
$O(n^2)$	orden cuadrático
$O(n^a)$	orden polinomial (a>2)
$O(a^n)$	orden exponencial (a>2)
$O(n!)$	orden factorial

Tabla B.2: Ordenes de algoritmos

NP-Hard: Es una clase que al menos es tan compleja como NP-Complete; se cumple que los problemas NP-Hard pueden no ser NP y no pueden ser problemas de decisión.

En la Figura B.1 se puede visualizar mejor la separación de clases. En el libro Johnson [75] se detalla en profundidad acerca de la complejidad en algoritmos.

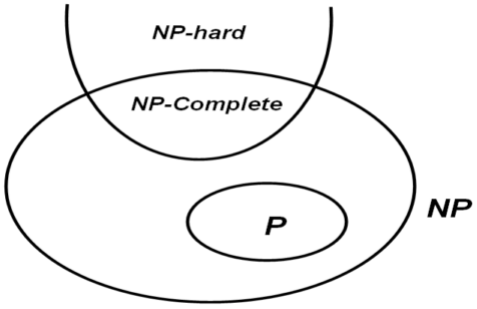


Figura B.1: Clases de complejidad (Fuente: Internet)

Se suelen utilizar los métodos exactos para resolver problemas sencillos y de orden polinomial; los métodos de aproximación se comportan mejor cuando se tratan de resolver problemas en la clase NP-Hard. A pesar de que difieren en su implementación, en muchas investigaciones se proponen modelos que combina lo mejor de ambas técnicas. En las siguientes sesiones se tratan en detalle estos dos métodos.

Métodos Exactos

Introducción

Como se dijo anteriormente, los Métodos Exactos tratan de obtener la mejor solución si esta existe. En general tratan de reducir el espacio de soluciones y el número de las diferentes alternativas para ser examinadas [74, p 15-16]. Existen dos clases principales, están los Métodos Constructivos y los Métodos de Enumeración [12, p. 192].

Los Métodos Constructivos exploran propiedades especificas del modelo del schedule para construir una solución que garantice ser óptima. Por ejemplo el algoritmo de Johnson (1954) es un caso de un algoritmo constructivo conocido por resolver el modelo Flow-Shop con 2 máquinas; el algoritmo de Lawler (1973) resuelve aplicando esta técnica el modele $1|prec|\gamma$, siendo γ una función objetivo maximizadora; por último el algoritmo de Morre (1968) resuelve el modelo $1||\sum U_j$.

Los Métodos de Enumeración garantizan que explícitamente o implícitamente se evaluarán todas las soluciones del modelo. En el caso implícito se suele llegar a un estado donde la solución ya no es factible y seguir analizando no generará una óptima, por lo tanto es proceso de análisis se detiene antes e implícitamente se descarta parte del espacio de solución. Las técnicas más usadas son:

Programación Dinámica

De ser posible, la idea consiste en dividir el problema en sub-problemas más sencillos con un enfoque recursivo. Cada uno de los sub-problema tiene asociado un estado, cuando dos o más soluciones parciales logran el mismo estado se recurre a un criterio de descarte, quedándose con un valor y eliminando el resto.

Existes dos modalidades denominadas Top-Down y Bottom-Up. La modalidad Top-Down divide el problema en sub-problemas y mantiene registro de resultados anteriores por si llegan a ser necesarios. El modo Bottom-Up

se resuelven todos los problemas necesarios de antemano y se utilizan para resolver problemas mayores.

Ejemplo 1. La sucesión de Fibonacci se define como:

$$Fib(n) = \begin{cases} 0 & \text{si } n = 0 \\ 1 & \text{si } n = 1 \\ Fib(n-1) + Fib(n-2) & \text{si } n > 1 \end{cases}$$

Una implementación recursiva se encuentra en el **Algorithm 6**, esta forma tiene complejidad $O(2^n)$ y correspondiente a la clase NP. Si se calcula $fib(5)$ los pasos requeridos son:

1. fib(5)
2. fib(4) + fib(3)
3. (fib(3) + fib(2)) + (fib(2) + fib(1))
4. ((fib(2) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))
5. (((fib(1) + fib(0)) + fib(1)) + (fib(1) + fib(0))) + ((fib(1) + fib(0)) + fib(1))

Se puede observar que para llegar al resultado muchos cálculos son repetidos por lo que no es muy eficiente. Llevándolo al enfoque de Programación Dinámica se obtiene el **Algorithm 7**; el orden de este algoritmo es $O(n)$ y más eficiente que $O(2^n)$. El enfoque empleado es el Top-Down debido que en las lineas 10-11 se utilizan los estados anteriores definidos en las lineas 7-8.

Algoritmo 6 Fibonacci recursivo

```
1: function FIB(n)
2:     sí n==0 or n==1 entonces
3:         retornar 1
4:     else
5:         retornar Fibonacci(n-1) + Fibonacci(n-2)
6:     fin
7: fin function
```

Algoritmo 7 Fibonacci - Programación Dinámica

```
1: function FIB2(n)
2:     i ← 2
3:     step1 ← 1
4:     step2 ← 1
5:     sí n ≥ 2 entonces
6:         mientras  i ≤ n hacer
7:             step1Tmp ← step1
8:             step2Tmp ← step2
9:
10:            step1 ← step2
11:            step2 ← step2Tmp + step1Tmp
12:
13:            i ← i + 1
14:        fin
15:        retornar step2
16:    else
17:        retornar 1
18:    fin
19: fin function
```

Branch-and-Bound

Es otra técnica de optimización donde se utiliza un límite superior e inferior para descartar soluciones y acotar el espacio de búsqueda a través del uso de atributos o propiedades [10, p 30]. Los nodos que no pertenecen dentro del límite no se exploran. Este algoritmo se forma por dos procesos: "branching" y 'bounding'. El branching se encarga de dividir un conjunto de soluciones candidatas en subconjuntos más pequeño; el bounding es el encargado de calcular los límites de un conjunto. El algoritmo culmina cuando la búsqueda encontró una solución o descartó a todas.

Branch-and-Cut

Es la técnica Branch-and-Bound con una etapa adicional llamada "cutting". El motivo es reducir más el espacio de búsqueda agregando criterios. Agregar más criterios puede mejorar la etapa de bounding y permite resolver sub-problemas sin utilizar branching.

Métodos de Aproximación

Introducción

Los métodos de aproximación pueden ser considerados como un proceso de optimización bajo consideraciones que se suponen adecuadas o de buena calidad. Se suelen aplicar cuando no existen métodos exactos o si los hay son muy costosos en consumo de recursos y tiempo [12, p 175]. En la industria existen muchos métodos de aproximación que encaran problemas de scheduling NP-Hard y debido a que están basados en conceptos e ideas generales permite combinar varios de ellos.

Las heurísticas y metaheurísticas son dos formas de clasificar a estos métodos. Según Zanakins y Evans (1981), "*un heurístico es un procedimiento simple, a menudo basado en el sentido común, que se supone que ofrecerá una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido*".

Los métodos de resolución mediante heurísticos se pueden clasificar como [76]:

Métodos constructivos: Construyen una solución definiendo diferentes partes de ella en sucesivos pasos.

Métodos de reducción: Buscan algún patrón o característica que permita simplificar el problema.

Métodos de descomposición: Se simplifica el problema dividiéndolo en sub-problemas más sencillos, la solución resulta de la solución de cada uno de los sub-problemas.

Métodos de manipulación del modelo: Obtienen una solución a partir de la solución de un modelo más simplificado.

Métodos de búsqueda por entornos: Se parte de una solución inicial e iterativamente se selecciona las soluciones vecinas más adecuadas.

Las metaheurísticas son otra formas de realizar búsquedas de manera sencilla y al igual que los heurísticos no garantizan una mejor solución. La diferencia radica en que las metaheurísticas evitan estancarse en soluciones locales y tratan de explorar todo el espacio. Las técnicas son frecuentemente utilizadas en problemas de optimización combinatoria, en particular en los problemas Flow-Shop, Open-Shop y Job-Shop.

La manera que las heurísticas o meta-heurísticas encuentran las soluciones es a través del pasaje de un estado a otro, a este proceso se le llama Búsqueda Local (BL). Durante este proceso la nueva solución se suele generar dentro del neighbourhood de la solución anterior. Un neighbourhood $N(x)$ de una solución x es un subconjunto del espacio de soluciones del problema conteniendo uno o más óptimos locales. La transacción de una solución x a una x' se realiza modificando atributos de la solución x, a esto se le llama Neighbourhood Move. Adicionando el uso de una función que pondera a la solución encontrada es posible determinar el óptimo global comparándola con las otras encontradas [74, p. 17].

La principal desventaja de la Búsqueda Local es que se suelen estancar en óptimos locales, no llegando a calcular el óptimo global, por lo que se suele agregar técnicas probabilistas y/o analogías de hechos reales para evitarlo (ver Figura B.2).

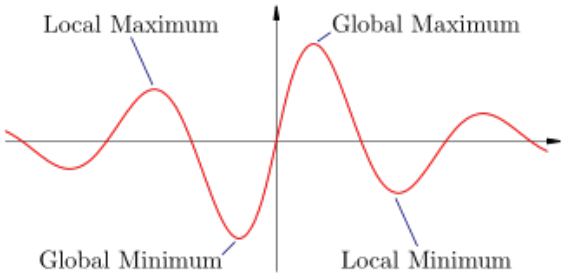


Figura B.2: Óptimos globales y locales (Fuente: Internet)

En los últimos años las técnicas más extendidas han sido los Algoritmos Genéticos (GA), Búsqueda Tabú, Simulated Annealing, Colonias de Hormigas, GRASP y las Redes Neuronales. En la Tabla B.3 referente a [76, p 113] se indica el porcentaje de artículos (en base a 340 trabajos) en los que fué utilizada una determinada metaheurística, bien de forma individual o en combinación con otra.

Todas ellas comparten las siguientes características [77]:

- No saben cuando llegan a una solución óptima.
- Son algoritmos aproximativos, por lo tanto no garantizan buena solución.
- La generación de soluciones malas determinan posibles caminos para la obtención de las buenas. "aprenden"del error.
- Practican el termino "pensamiento en masa o colectivo".
- Son sencillos y generales.
- Su comportamiento es determinado por los parámetros del problema.
- Flexibilidad a la hora de combinarse con otras metaheurísticas.

Aunque cada una de ellas están implementadas de forma distinta o siguen un concepto base, muchos trabajos realizados recientemente han dado sus frutos producto de la combinación de meta-heurísticas. En [78] (2011) se analiza la calidad de la solución de un problema de scheduling explorando el algoritmo PSO (Particle Swarm Optimization) junto a Taboo Search y Simulated Annealing. Tang y colegas [79] (2011) proponen un algoritmo híbrido haciendo uso de Particle Swarm Optimization (PSO) y Algoritmos Genéticos para la resolución del problema flexible job-shop scheduling; como resultado de los experimentos realizados probaron que el algoritmo resultó tan eficiente como los implementados a la fecha.

Es importante ver que el tipo de problema define el modo que se debe explorar y analizar las distintas técnicas para obtener un buen resultado y por lo general implica "jugar"con los parámetros "calidad de solución", "tiempo disponible"y "dimensión"del problema.

En la siguiente secciones se tratará en más detalle algunas de las metaheurísticas más utilizadas.

Algoritmos Genéticos (GA)

Introducidos en la década de los 70 por Holland (1975) y luego por Goldberg (1988), están inspirados en la evolución de los seres vivos y suelen utilizarse para la resolución de problemas combinatorios. Son muy usados en el marco de Shop Scheduling debido a que se pueden explorar grandes espacios de soluciones y se puede extender fácilmente con el uso de otras técnicas.

Debido a que está basado en la evolución biológica se manejan conceptos de herencia, cruzamiento (crossover), mutación (mutation), selección (selection), selección natural y sobrevivencia de los mejores adaptados. La "población" es lo que se denomina conjunto de soluciones; en cada paso del algoritmo se genera una nueva

Metaheurística	Sigla en inglés	Cantidad	Porcentaje
Algoritmos genéticos / Algoritmos evolutivos	GA / EA	92	21,45
Búsqueda tabú	TS	76	17,72
Optimización por colonia de hormigas	ACO	51	11,89
Simulated Annealing	SA	50	11,66
Procesos aleatorizados y adaptativos de búsqueda voraz	GRASP	40	9,32
Búsqueda de vecindad variable	VNS	29	6,76
Búsqueda local	LS	19	4,43
Optimización por enjambre de partículas	PSO	17	3,96
Búsqueda dispersa	SS	16	3,73
Colonia artificial de abejas	ABC	10	2,33
Redes neuronales artificiales	ANN	8	1,86
Búsqueda armónica	HS	5	1,17
Algoritmos meméticos	MA	4	0,93
Algoritmo de la luciérnaga	FF	3	0,70
Otras metaheurísticas		9	2,10

Tabla B.3: Número de artículos publicados en la literatura entre 2003 y 2012.

población denominada "generación". Cada individuo de la población es evaluada y se le asigna un peso (fitness value) que se relaciona con la función objetivo a optimizar (ej: costo, makespan, etc.), cuando mejor sea el valor de la función objetivo mayor será el fitness value.

En el Algoritmo 8 se muestra la estructura base del GA. En las lineas 2 y 3 se inicia la población inicial; en este paso se suele ingresar a la población soluciones "cercanas" a las que se esperan con el objetivo de orientar la búsqueda y reducir tiempo de cómputos. En la linea 4 se puede ver la utilización de un criterio de parada; este criterio puede ser que la diferencia entre la mejor solución actual y otra cualquiera sea menor a un determinado valor, por ejemplo 0,005. Esto significa el error dispuesto a tolerar en la solución. En la linea 5 se asigna un valor de fitness a cada individuo; en la 6 se seleccionan candidatos adecuados para la nueva generación; en la 7 se genera un conjunto de descendientes a partir de los individuos seleccionados, mediante operadores que emula la evolución natural (cruzamiento, mutación, selección natural), por último en la linea 8 se cuenta con un mecanismo que realiza el recambio generacional.

Algoritmo 8 Algoritmo Genético

```
1: Salida: Población incluyendo mejor solución y valor de la función objetivo
2: generación ← 0
3: Iniciar aleatoriamente la población inicial P(0)
4: mientras (no CriterioParada) hacer
5:         evaluar(P(generación))
6:         padres ← seleccionar(P(generación))
7:         hijos ← aplicar operadores evolutivos(padres)
8:         nueva población ← reemplazar(hijos, P(generación))
9:         generación = generación + 1
10:        P(generación) = nueva población
11: fin
12: retornar mejor solución encontrada
```

El trabajo realizado por Di-Wei Huan y Jimmy Lin [80] se pueden ver las ventajas de exploración de soluciones e integración con otras técnicas; aquí GA se implementa para poder correr en clusteres y junto al framework map-reduce es posible escalar la población de soluciones en el orden de 10^7. José Fernando, Jorge José y Mauricio G[81] propusieron un híbrido para el problema de Job Shop Scheduling, donde el schedule es construido está combinando con la metaheurística Priority Rules, en el cual las prioridades están definidas por el algoritmo genético.

Simulated Annealing (SA)

Simulated Annealing es otra de las muchas heurísticas diseñadas para dar una buena aunque no óptima solución en la resolución de problemas combinatorios. Fué inicialmente propuesto por Kirkpatrick (1983) y Cerny (1985), desde entonces ha sido aplicado en el procesamiento de imágenes, física molecular, química y job-shop scheduling [82, p. 3], entre otras aplicaciones.

Su comportamiento está basado en la analogía del recocido del acero y cerámicas; está técnica consiste en calentar y luego enfriar lentamente el material para variar sus propiedades físicas. El calor causa que los átomos aumenten su energía y que puedan así desplazarse de sus posiciones iniciales (un mínimo local de energía); el enfriamiento lento les da mayores probabilidades de recristalizar en configuraciones con menor energía que la inicial (mínimo global) (fuente Wikipedia).

SA es una extensión de Búsqueda Local y como se nombró anteriormente esta técnica corre el riesgo de estancarse en los óptimos locales y quedar lejos del global. La forma que SA evita este problema es a través de los neighbourhood moves, con una variable probabilista se decide si aplicar o no el movimiento. La probabilidad de aceptar un movimiento genera un desplazamiento φ en la función objetivo f a optimizar. Esta probabilidad se denomina función de aceptación y es igual a $exp(-\varphi/T)$, siendo T un parámetro de control que corresponde a la temperatura en la analogía. Esto significa que para valores altos de φ la probabilidad de ser aceptado serán altas; en el caso de T si el valor es grande los movimientos serán aceptado la mayoría de las veces, y a medida que T se acerca a cero los movimientos serán menos aceptados [82, p. 3].

El algoritmo comienza con un valor T relativo alto para evitar estancamiento en óptimos locales inicialmente. Por cada estado que la temperatura T pasa se realizan neighbourhood moves como medio de exploración de soluciones. En el Algoritmo 9 se muestra un pseudocódigo de SA.

Algoritmo 9 Simulated Annealing [82]

```
1:  Seleccionar estado inicial i ∈ S
2:  t ← 0
3:  n ← 0
4:  Repetir
5:      Seleccionar temperatura inicial T > 0
6:      Repetir
7:          Generar estado j, neighbourhood de i
8:          Calcular φ ← f(j) − f(i)
9:          Si φ < 0 entonces i ← j
10:         Si random(0, 1) < exp(−φ/T) entonces i ← j
11:         n ← n + 1
12:     Hasta nigualN
13:     t ← t + 1
14:     T ← T(i)
15: Hasta CriterioDeParada
```

Algoritmo de Búsqueda Tabú

La Búsqueda Tabú tuvo origen a finales de la década de los 70 por Fred Glober, esta meta-heurística es otra clase de los algoritmos de Búsqueda Local y enfrenta el mismo problema que Simulated Annealing: no estancarse en los óptimos locales. Su nombre está basado en la definición de Tabú: "*refiere a conductas o acciones prohibidas por la sociedad debido a cuestiones culturales, sociales o religiosas*". Para el caso de esta técnica, los tabúes son prohibiciones de soluciones que se realizan durante el proceso de búsqueda y evitan el riesgo de quedar atrapado en un conjunto de soluciones [77, p. 5].

Esta metaheurística obtiene mejor rendimiento que la Búsqueda Local debido a que utiliza una memoria como estructura de datos. Esta memoria es principalmente utilizada para guiar la búsqueda, y a diferencia de Simulated Annealing, guardar una mala decisión es más preferible que una elección aleatoria.

La idea detrás es comenzar con una solución y utilizar neighbourhood moves para explorar soluciones vecinas,

se selecciona la mejor entre ellas y a la vez se van generando soluciones "prohibidas". Se debe tener en cuenta que las prohibiciones pueden generar regiones del espacio de soluciones sin explorar, la medida es introducir los "criterios de aceptación", o sea, se aceptan soluciones tabú si cumplen algún criterio [77, p. 6-7]. El proceso continua y se detiene definiendo algún criterio de parada.

Esta heurística ha sido utilizada en varios trabajos a los largo de los años por su eficiencia computacional y calidad de soluciones. Por ejemplo en el artículo publicado por D. Eric [83], su algoritmo planteado prueba ser más eficiente en resolver Job-Shop Scheduling que con Simulated Annealing. Schmidt y Keith [84] proponen una implementación de Job-Shop Scheduling con Sequence Dependent Setup Times aplicando Búsqueda Tabú.

Algoritmo 10 Algoritmo Búsqueda Tabú (Wikipedia)

```
1: sBest ← s0
2: tabuList ← []
3: mientras not criterioDeParada hacer
4:     candidateList ← []
5:     bestCandidate ← null
6:     para cada sCandidate en sNeighborhood hacer
7:         sí (not tabuList.contain(sCandidate) y fitness(sCandidate)>fitness(bestCandidate)) entonces
8:             bestCandidate ← sCandidate
9:         fin
10:    fin
11:    sí fitness(bestCandidate) > fitness(sBest) entonces
12:        sBest ← bestCandidate
13:    fin
14:    Agregar a tabuList bestCandidate
15:    sí tabuList.size > maxTabuSize entonces
16:        tabuList.removeFirst
17:    fin
18: fin
19: retornar sBest
```

Greedy randomized adaptive search procedure (GRASP)

GRASP es otro algoritmo metaheurístico desarrollado por Feo y Resende en 1995, utilizado para la resolución de problemas combinatorios, en particular a los de scheduling. Una de las ventajas que tiene respecto a otras heurísticas es que solo hay dos parámetros a configurar: la cantidad de iteraciones y el tamaño de la lista de candidatos.

Como se puede ver en el Algoritmo 11, GRASP es iterativo y se forma por dos fases. La primera etapa denominada fase constructiva (ver Algoritmo 12) se forma en cada iteracion una lista de elementos candidatos que pueden ser incluidos en la solución que se está construyendo. Cada uno de estos candidatos son elegidos de manera que no se pierda la factibilidad de la solución, posteriormente se elige aleatoriamente uno de ellos y se evalúa el costo incremental.

La segunda fase denominada Búsqueda Local, tiene como objetivo mejorar la solución construida en la fase anterior. Como se nombró anteriormente sobre la Búsqueda Local, la idea es partir de una solución y a través de neighbourhood moves explorar soluciones vecinas. Esta fase culmina cuando se encuentra una buena solución en el vecindario.

Esta técnica ha sido empleada para resolver el problema de Job-Shop Scheduling [85, p. 28]. En el trabajo realizado por Binato y colaboradores en 2002, se incorpora al algoritmo dos conceptos nuevos: una estrategia de intensificación y una aproximación al principio de optimalidad en la fase de construcción. Mediante la aplicación de estas variantes se presenta una gran mejora respecto a resultados obtenidos por el algoritmo GRASP clásico.

Algoritmo 11 GRASP pseudocodigo [86, p. 2]

```
1: procedimiento GRASP( maxIterations, seed)
2:     para cada k=1,..., maxIterations  hacer
3:         solution ← GreedyRandomizedConstruction(seed)
4:         solution ← LocalSearch(solution)
5:         updateSolution(solution, bestSolution)
6:     fin
7:     retornar bestSolution
8: fin
```

Algoritmo 12 GRASP, Fase constructiva [86, p. 2]

```
procedimiento GreedyRandomizedConstruction(seed)
    solution ← ∅
    Evaluar los costos incrementales de los elementos candidatos
    mientras solution no sea una completa solución hacer
        Construir la lista de candidatos restringidos (RCL)
        Seleccionar un s aleatorio de RCL
        solution ← solution ∪ {s}
        Evaluar costo incremental
    fin
    retornar solution
fin
```

Ant Colony Optimization (ACO)

Ant Colony Optimization es otro algoritmo que combina búsqueda local, dispatching rules (Reglas de despacho) y otras técnicas probabilística. Inicialmente propuesto por Marco Dorigo en 1992 y al igual que Algoritmos Genéticos se basa en el funcionamiento y comportamiento biológico de los seres vivos [87, p. 387].

ACO se inspira en el comportamiento de las colonias de hormigas; cuando una hormiga recorre un camino para llegar a un destino vá dejando a lo largo un químico como señal para que otras hormigas la sigan. A nivel de algoritmo una colonia de hormigas artificial construye iterativamente utilizando los rastros artificiales de otras hormigas, esto se refleja a utilizar soluciones previamente halladas. La cantidad de sustancia que se deja a lo largo del camino es una forma de comunicación en la colonia; a nivel de metaheurística da un contexto más global de las soluciones. Dado que la metaheurística sigue un "pensamiento colectivo"el óptimo encontrado no puede ser un óptimo local; sin embargo ACO se puede extender para mejorar la búsqueda local, en [88, p. 92] se propone una implementación.

En el libro de Marco Dorigo [88, p.174], se trata el tema de scheduling para distintos modelos, entre ellos Job-Shop, Open-Shop y Flow-Shop, donde tomando como referencia el pseudocodigo en Algorithm 13 se propone una implementación para cada etapa.

Algoritmo 13 Ant colony Optimization [87, p. 387]

1: **Inicialización**: setear parámetros e iniciar caminos de feromona
2: **Generar soluciones**: Generar l soluciones usando combinaciones de caminos de feromona y Dispatching Rules.
3: **Mejorar soluciones**: Aplicar Búsqueda Local a cada una de las l soluciones.
4: **Actualizar configuración**: Si se encontró una mejor solución en las etapas anteriores, actualizar la mejor solución global. Actualizar caminos de feromona y retornar a la etapa "Generar soluciones".

Particle Swarm Optimization

Particle Swarm Optimization (PSO) es un método de búsqueda y optimización desarrollado por Eberhart y Kennedy en 1995, basado en el comportamiento colectivo de las aves o de los peses. PSO es un método

probabilístico y abstracto por lo que se han propuesto muchas implementaciones para mejorar velocidad de convergencia y calidad de solución.

La idea se basa en el comportamiento que tiene una manada sin líder; al momento de buscar alimento, cada miembro explora el territorio de forma aleatoria, cuando uno de ellos encuentra la fuente de alimento el resto de los miembros lo siguen. Los autores originales iniciaron este concepto desde una perspectiva social: los individuos que integran una sociedad tienen una opinión influenciada por la creencia global. Cada individuo puede modificar su opinión (estado) dependiendo de tres factores [89, p. 7]: conocimiento del entorno (su adaptación), los estados de la historia por la que ha pasado el individuo (memoria) y estados en la historia de los individuos cercanos (memoria del vecindario).

El proceso de búsqueda de una solución óptima en el algoritmo PSO sigue este mismo comportamiento. Cada individuo denominado partícula forma parte de un enjambre que representa el entorno social. Cada partícula tiene asociada dos propiedades: una posición en el espacio de búsqueda y una velocidad a la que se desplaza. Cada propiedad es representada por una ecuación (ver x_i y v_i en Algorithm 14). La posición de la partícula es la composición de la velocidad inicial y dos valores ponderados aleatoriamente, uno representa la tendencia de preservar su estado anterior y el otro la tendencia de preservar un estado global ($pBest$ y $gBest$ respectivamente).

Esta metaheurística tiene como ventaja de capturas el contexto local y global para hallar una solución, sin embargo como se vé en el Algorithm 14, hay parámetros que influyen en el comportamiento de cada individuo, por lo que se requiere de un proceso de prueba para lograr un mejor resultado en función del problema a encarar.

Algoritmo 14 PSO Global [89, p. 23]

1: $S \leftarrow InicializarCumulo()$
2: **mientras** no se alcance condición de parada **hacer**
3: **para cada** $i = 1$ to $size(S)$ **hacer**
4: evaluar cada partícula x_i de S
5: **sí** $fitness(x_i)$ es mejor que $fitness(pBest_i)$ **entonces**
6: $pBest_i \leftarrow x_i$
7: $fitness(x_i) \leftarrow fitness(pBest_i)$
8: **fin**
9: **sí** $fitness(pBest_i)$ es mejor que $fitness(gBest)$ **entonces**
10: $gBest \leftarrow pBest_i$
11: $fitness(gBest) \leftarrow fitness(pBest_i)$
12: **fin**
13: **fin**
14: **para cada** $i = 1$ to $size(S)$ **hacer**
15: $v_i \leftarrow w.v_i + \varphi_1.rand_1.(pBest_i - x_i) + \varphi_2.rand_2.(gBest - x_i)$
16: $x_i = x_i + v_i$
17: **fin**
18: **fin**
19: **retornar** $gBest$

Conclusión

Las metaheurísticas aplicadas a los problemas de optimización combinatoria es un campo en pleno fortalecimiento y desarrollo. Nuevos modelos y estrategias extienden los ya existentes, donde también surgen nuevas propuestas debido a la demanda práctica. Como punto fuerte, la flexibilizad que presentan ha producido como tendencia actual la combinación de varias de las técnicas mencionadas, permitiendo sacar lo mejor de cada en las implementaciones realizadas. Entre los factores como la limitación de poder de cómputo, el poco tiempo disponible, entre otros, hace que sea factible el estudio y desarrollo de algorítmicos y técnicas.

ANEXO C

Tablas

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1
demanda	dm1=132, dm2=202, dm3=445, dm4=28, dm5=441
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.1: Escenario 1

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=2, m2=2, m3=2, m4=2, m5=2
demanda	dm1=132, dm2=202, dm3=445, dm4=28, dm5=441
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.2: Escenario 2

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1
demanda	dm1=132, dm2=202, dm3=445, dm4=28, dm5=441
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.3: Escenario 3

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	12 (h1...h12)
#moldes	9 (m1...m9)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1, m6=1, m7=1, m8=1, m9=1
demanda	dm1=132, dm2=202, dm3=405, dm4=28, dm5=441, dm6=508, dm7=85, dm8=70, dm9=43
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6, tc6=60.6, tc7=60.6, tc8=60.6, tc9=60.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18, tv6=40, tv7=42, tv8=53, tv9=55
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2, tq6=44.9, tq7=44.9, tq8=44.9, tq9=44.9
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$, $\{m6, m7\}$, $\{m8, m9\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\}x\{h1, h2, h3, h4, h5, h6, h7\}$, $\{m6, m7\}x\{h8, h9, h10\}$, $\{m8, m9\}x\{h11, h12\}$

Tabla C.4: Escenario 4

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	12 (h1...h12)
#moldes	9 (m1...m9)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1, m6=1, m7=1, m8=1, m9=1
demanda	dm1=132, dm2=202, dm3=405, dm4=28, dm5=441, dm6=508, dm7=85, dm8=70, dm9=43
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6, tc6=60.6, tc7=60.6, tc8=60.6, tc9=60.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18, tv6=40, tv7=42, tv8=53, tv9=55
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2, tq6=44.9, tq7=44.9, tq8=44.9, tq9=44.9
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$, $\{m6, m7\}$, $\{m8, m9\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$, $\{m6, m7\} \times \{h8, h9, h10\}$, $\{m8, m9\} \times \{h11, h12\}$

Tabla C.5: Escenario 5

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1
demanda	dm1=660, dm2=1010, dm3=2225, dm4=138, dm5=2203
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.6: Escenario 6

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=2, m2=2, m3=2, m4=2, m5=2
demanda	dm1=660, dm2=1010, dm3=2225, dm4=138, dm5=2203
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.7: Escenario 7

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	9 (m1...m9)
#moldes disponibles	m1=1, m2=1, m3=1, m4=1, m5=1, m6=1, m7=1, m8=1, m9=1
demanda	dm1=660, dm2=1010, dm3=2225, dm4=138, dm5=2203, dm6=2541, dm7=423, dm8=350, dm9=215
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6, tc6=60.6, tc7=60.6, tc8=60.6, tc9=60.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18, tv6=40, tv7=42, tv8=53, tv9=55
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2, tq6=44.9, tq7=44.9, tq8=44.9, tq9=44.9
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$, $\{m6, m7, m8, m9\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$, $\{m6, m7\} \times \{h8, h9, h10\}$, $\{m8, m9\} \times \{h11, h12\}$

Tabla C.8: Escenario 8

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	9 (m1...m9)
#moldes disponibles	m1=2, m2=2, m3=2, m4=2, m5=2, m6=2, m7=2, m8=1, m9=2
demanda	dm1=660, dm2=1010, dm3=2225, dm4=138, dm5=2203, dm6=2541, dm7=423, dm8=350, dm9=215
piezas compartidas	si (p1)
#piezas compartidas	p1=1
requerimientos de piezas	m1 (p1), m2 (p1)
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6, tc6=60.6, tc7=60.6, tc8=60.6, tc9=60.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18, tv6=40, tv7=42, tv8=53, tv9=55
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2, tq6=44.9, tq7=44.9, tq8=44.9, tq9=44.9
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$, $\{m6, m7, m8, m9\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$, $\{m6, m7\} \times \{h8, h9, h10\}$, $\{m8, m9\} \times \{h11, h12\}$

Tabla C.9: Escenario 9

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=2, m2=2, m3=2, m4=2, m5=2
demanda	dm1=6000, dm2=5000, dm3=4000, dm4=4000, dm5=2000
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.10: Escenario 10

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	7 (h1...h7)
#moldes	5 (m1...m5)
#moldes disponibles	m1=10, m2=10, m3=10, m4=15, m5=15
demanda	dm1=6000, dm2=5000, dm3=4000, dm4=4000, dm5=2000
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2
compatibilidad molde-molde	$\{m1, m2, m3, m4, m5\}$
compatibilidad molde-heater	$\{m1, m2, m3, m4, m5\} \times \{h1, h2, h3, h4, h5, h6, h7\}$

Tabla C.11: Escenario 11

Parámetro	Valor
duración de jornada laboral (min)	1440
#heaters	$\{i \mid i \geq 5 \wedge i \leq 50 \wedge i \,\%5 == 0\}$
#moldes	9 (m1...m9)
#moldes disponibles	m1=20, m2=20, m3=20, m4=20, m5=20, m6=20, m7=20, m8=20, m9=20
demanda	dm1=6000000, dm2=5000000, dm3=4500000, dm4=5000000, dm5=2800000, dm6=6600000, dm7=1000000, dm8=7500000, dm9=6000000
piezas compartidas	no
tiempo de colocado (min)	tc1=66.8, tc2=66.8, tc3=41.6, tc4=41.6, tc5=41.6, tc6=60.6, tc7=60.6, tc8=60.6, tc9=60.6
tiempo de vulcanizado (min)	tv1=12.5, tv2=12.5, tv3=30, tv4=26, tv5=18, tv6=40, tv7=42, tv8=53, tv9=55
tiempo de quitado (min)	tq1=62.2, tq2=62.2, tq3=25.2, tq4=25.2, tq5=25.2, tq6=44.9, tq7=44.9, tq8=44.9, tq9=44.9
compatibilidad molde-molde	$\{m1, ..., m9\}$
compatibilidad molde-heater	Todos los moldes se puede procesar en cualquier heater

Tabla C.12: Escenario de estrés

Scene1

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	11	0.0086	11	0.0127	12	12.0000	0.0000	0.0941	11	11.0000	0.0000	0.0942	15	0.0010	15	0.0011	11	11.0000	0.0000	0.0142	11	11.0000	0.0000	0.0143
Caso 2	10	0.0006	10	0.0008	13	13.0000	0.0000	0.0510	10	10.0000	0.0000	0.0511	14	0.0001	14	0.0002	10	10.0000	0.0000	0.0118	10	10.0000	0.0000	0.0118
Caso 3	9	0.0005	9	0.0010	11	11.0000	0.0000	0.0165	9	9.0000	0.0000	0.0166	12	0.0001	9	0.0004	9	9.0000	0.0000	0.0199	9	9.0000	0.0000	0.0200
Caso 4	10	0.0005	10	0.0008	10	10.0000	0.0000	0.0119	10	10.0000	0.0000	0.0120	13	0.0003	13	0.0003	10	10.0000	0.0000	0.0132	10	10.0000	0.0000	0.0133
Caso 5	11	0.0006	11	0.0008	13	13.0000	0.0000	0.0110	11	11.0000	0.0000	0.0111	15	0.0001	15	0.0001	11	11.0000	0.0000	0.0133	11	11.0000	0.0000	0.0134
Caso 6	12	0.0007	12	0.0009	14	14.0000	0.0000	0.0118	12	12.0000	0.0000	0.0119	17	0.0001	17	0.0002	12	12.0000	0.0000	0.0101	12	12.0000	0.0000	0.0102
Caso 7	10	0.0008	10	0.0017	13	13.0000	0.0000	0.0148	10	10.0000	0.0000	0.0149	13	0.0001	10	0.0002	10	10.0000	0.0000	0.0115	10	10.0000	0.0000	0.0115
Caso 8	9	0.0010	9	0.0012	11	11.0000	0.0000	0.0170	9	9.0000	0.0000	0.0171	12	0.0001	9	0.0002	9	9.0000	0.0000	0.0089	9	9.0000	0.0000	0.0090
Caso 9	10	0.0006	10	0.0009	12	12.0000	0.0000	0.0140	10	10.0000	0.0000	0.0141	13	0.0000	10	0.0001	10	10.0000	0.0000	0.0089	10	10.0000	0.0000	0.0090
Caso 10	8	0.0005	8	0.0007	10	10.0000	0.0000	0.0145	8	8.0000	0.0000	0.0146	11	0.0001	8	0.0001	8	8.0000	0.0000	0.0081	8	8.0000	0.0000	0.0082

Scene2

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	5	0.0006	5	0.0035	5	5.0000	0.0000	0.0619	5	5.0000	0.0000	0.0620	5	0.0001	5	0.0002	5	5.0000	0.0000	0.0176	5	5.0000	0.0000	0.0177
Caso 2	7	0.0003	7	0.0005	7	7.0000	0.0000	0.0611	6	6.0000	0.0000	0.0613	7	0.0000	7	0.0001	7	7.0000	0.0000	0.0220	6	6.0000	0.0000	0.0221
Caso 3	5	0.0002	5	0.0005	5	5.0000	0.0000	0.0266	5	5.0000	0.0000	0.0267	5	0.0000	5	0.0001	5	5.0000	0.0000	0.0230	5	5.0000	0.0000	0.0232
Caso 4	6	0.0002	6	0.0005	5	5.0000	0.0000	0.0311	5	5.0000	0.0000	0.0312	6	0.0001	6	0.0001	5	5.0000	0.0000	0.0241	5	5.0000	0.0000	0.0242
Caso 5	5	0.0002	5	0.0004	5	5.0000	0.0000	0.0234	5	5.0000	0.0000	0.0235	5	0.0001	5	0.0002	5	5.0000	0.0000	0.0205	5	5.0000	0.0000	0.0206
Caso 6	6	0.0001	6	0.0005	6	6.0000	0.0000	0.0242	5	5.0000	0.0000	0.0243	6	0.0001	6	0.0001	6	6.0000	0.0000	0.0251	5	5.0000	0.0000	0.0253
Caso 7	5	0.0002	5	0.0005	5	5.0000	0.0000	0.0188	5	5.0000	0.0000	0.0189	5	0.0001	5	0.0002	5	5.0000	0.0000	0.0205	5	5.0000	0.0000	0.0206
Caso 8	5	0.0002	5	0.0005	5	5.0000	0.0000	0.0237	5	5.0000	0.0000	0.0239	5	0.0001	5	0.0001	5	5.0000	0.0000	0.0227	5	5.0000	0.0000	0.0228
Caso 9	6	0.0002	6	0.0005	6	6.0000	0.0000	0.0290	5	5.0000	0.0000	0.0292	6	0.0001	6	0.0001	6	6.0000	0.0000	0.0274	5	5.0000	0.0000	0.0275
Caso 10	6	0.0002	6	0.0005	6	6.0000	0.0000	0.0243	5	5.0000	0.0000	0.0245	6	0.0000	6	0.0001	6	6.0000	0.0000	0.0254	5	5.0000	0.0000	0.0255

Scene3

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	11	0.0058	11	0.0059	12	12.0000	0.0000	0.0359	11	11.0000	0.0000	0.0361	12	0.0002	12	0.0002	11	11.0000	0.0000	0.0129	11	11.0000	0.0000	0.0129
Caso 2	10	0.0004	10	0.0004	12	12.0000	0.0000	0.0149	10	10.0000	0.0000	0.0150	13	0.0001	13	0.0001	10	10.0000	0.0000	0.0110	10	10.0000	0.0000	0.0110
Caso 3	11	0.0003	11	0.0004	13	13.0000	0.0000	0.0117	11	11.0000	0.0000	0.0117	14	0.0000	14	0.0001	11	11.0000	0.0000	0.0127	11	11.0000	0.0000	0.0128
Caso 4	10	0.0005	10	0.0005	12	12.0000	0.0000	0.0111	10	10.0000	0.0000	0.0111	13	0.0000	13	0.0001	10	10.0000	0.0000	0.0103	10	10.0000	0.0000	0.0104
Caso 5	8	0.0003	8	0.0003	10	10.0000	0.0000	0.0111	8	8.0000	0.0000	0.0111	11	0.0000	8	0.0002	8	8.0000	0.0000	0.0115	8	8.0000	0.0000	0.0116
Caso 6	10	0.0003	10	0.0003	12	12.0000	0.0000	0.0113	10	10.0000	0.0000	0.0114	14	0.0000	14	0.0001	10	10.0000	0.0000	0.0110	10	10.0000	0.0000	0.0110
Caso 7	9	0.0003	9	0.0003	11	11.0000	0.0000	0.0136	9	9.0000	0.0000	0.0136	12	0.0000	9	0.0001	9	9.0000	0.0000	0.0174	9	9.0000	0.0000	0.0175
Caso 8	11	0.0002	11	0.0003	13	13.0000	0.0000	0.0116	11	11.0000	0.0000	0.0117	13	0.0001	13	0.0003	11	11.0000	0.0000	0.0163	11	11.0000	0.0000	0.0164
Caso 9	11	0.0003	11	0.0003	11	11.0000	0.0000	0.0133	11	11.0000	0.0000	0.0134	13	0.0001	13	0.0002	11	11.0000	0.0000	0.0142	11	11.0000	0.0000	0.0143
Caso 10	10	0.0001	10	0.0001	12	12.0000	0.0000	0.0145	10	10.0000	0.0000	0.0146	13	0.0000	10	0.0002	10	10.0000	0.0000	0.0146	10	10.0000	0.0000	0.0146

Scene4

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	17	0.0001	17	0.0002	20	20.0000	0.0000	0.0322	17	17.0000	0.0000	0.0323	20	0.0001	17	0.0003	17	17.0000	0.0000	0.0263	17	17.0000	0.0000	0.0264
Caso 2	15	0.0001	15	0.0002	18	18.0000	0.0000	0.0253	15	15.0000	0.0000	0.0255	18	0.0001	15	0.0004	15	15.0000	0.0000	0.0241	15	15.0000	0.0000	0.0243
Caso 3	16	0.0001	16	0.0001	19	19.0000	0.0000	0.0244	16	16.0000	0.0000	0.0245	19	0.0001	16	0.0003	16	16.0000	0.0000	0.0265	16	16.0000	0.0000	0.0267
Caso 4	13	0.0001	13	0.0001	16	16.0000	0.0000	0.0254	13	13.0000	0.0000	0.0256	16	0.0001	13	0.0003	13	13.0000	0.0000	0.0243	13	13.0000	0.0000	0.0244
Caso 5	17	0.0001	17	0.0001	20	20.0000	0.0000	0.0244	17	17.0000	0.0000	0.0245	20	0.0001	17	0.0003	17	17.0000	0.0000	0.0245	17	17.0000	0.0000	0.0247
Caso 6	13	0.0001	13	0.0001	16	16.0000	0.0000	0.0248	13	13.0000	0.0000	0.0250	16	0.0001	14	0.0003	13	13.0000	0.0000	0.0271	13	13.0000	0.0000	0.0273
Caso 7	15	0.0001	15	0.0001	18	18.0000	0.0000	0.0233	15	15.0000	0.0000	0.0235	18	0.0000	15	0.0004	15	15.0000	0.0000	0.0233	15	15.0000	0.0000	0.0234
Caso 8	14	0.0002	14	0.0002	17	17.0000	0.0000	0.0233	14	14.0000	0.0000	0.0235	17	0.0001	15	0.0002	14	14.0000	0.0000	0.0231	14	14.0000	0.0000	0.0233
Caso 9	13	0.0003	13	0.0004	16	16.0000	0.0000	0.0252	13	13.0000	0.0000	0.0254	16	0.0001	14	0.0003	13	13.0000	0.0000	0.0250	13	13.0000	0.0000	0.0252
Caso 10	14	0.0001	14	0.0001	17	17.0000	0.0000	0.0243	14	14.0000	0.0000	0.0244	17	0.0000	14	0.0003	14	14.0000	0.0000	0.0230	14	14.0000	0.0000	0.0231

Figura C.1: Escenarios 1 a 4

Scene5

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	14	0.0001	14	0.0001	17	17.0000	0.0000	0.0529	14	14.0000	0.0000	0.0531	17	0.0001	14	0.0003	14	14.0000	0.0000	0.0394	14	14.0000	0.0000	0.0397
Caso 2	17	0.0001	17	0.0001	20	20.0000	0.0000	0.0383	17	17.0000	0.0000	0.0386	20	0.0001	17	0.0003	17	17.0000	0.0000	0.0358	17	17.0000	0.0000	0.0360
Caso 3	12	0.0001	12	0.0001	15	15.0000	0.0000	0.0365	12	12.0000	0.0000	0.0367	15	0.0001	15	0.0004	12	12.0000	0.0000	0.0325	12	12.0000	0.0000	0.0327
Caso 4	12	0.0003	12	0.0004	15	15.0000	0.0000	0.0366	12	12.0000	0.0000	0.0369	15	0.0001	12	0.0003	12	12.0000	0.0000	0.0353	12	12.0000	0.0000	0.0356
Caso 5	12	0.0001	12	0.0001	15	15.0000	0.0000	0.0369	12	12.0000	0.0000	0.0371	15	0.0001	12	0.0007	12	12.0000	0.0000	0.0360	12	12.0000	0.0000	0.0363
Caso 6	15	0.0001	15	0.0001	18	18.0000	0.0000	0.0352	15	15.0000	0.0000	0.0354	18	0.0001	15	0.0004	15	15.0000	0.0000	0.0361	15	15.0000	0.0000	0.0364
Caso 7	12	0.0001	12	0.0001	15	15.0000	0.0000	0.0323	13	13.0000	0.0000	0.0325	15	0.0001	13	0.0005	12	12.0000	0.0000	0.0336	12	12.0000	0.0000	0.0338
Caso 8	17	0.0001	17	0.0001	20	20.0000	0.0000	0.0361	18	18.0000	0.0000	0.0363	20	0.0001	18	0.0004	17	17.0000	0.0000	0.0404	17	17.0000	0.0000	0.0406
Caso 9	17	0.0001	17	0.0001	20	20.0000	0.0000	0.0343	17	17.0000	0.0000	0.0346	20	0.0002	17	0.0005	17	17.0000	0.0000	0.0372	17	17.0000	0.0000	0.0374
Caso 10	14	0.0001	14	0.0001	17	17.0000	0.0000	0.0394	14	14.0000	0.0000	0.0397	17	0.0001	14	0.0004	14	14.0000	0.0000	0.0383	14	14.0000	0.0000	0.0386

Scene6

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	50	0.0001	50	0.0001	50	50.0000	0.0000	0.0140	50	50.0000	0.0000	0.0141	65	0.0001	65	0.0001	50	50.0000	0.0000	0.0132	50	50.0000	0.0000	0.0133
Caso 2	39	0.0001	39	0.0001	48	48.0000	0.0000	0.0144	39	39.0000	0.0000	0.0145	51	0.0000	39	0.0002	39	39.0000	0.0000	0.0156	39	39.0000	0.0000	0.0156
Caso 3	48	0.0001	48	0.0001	51	51.0000	0.0000	0.0158	48	48.0000	0.0000	0.0159	74	0.0000	74	0.0002	48	48.0000	0.0000	0.0121	48	48.0000	0.0000	0.0121
Caso 4	55	0.0000	55	0.0001	63	63.0000	0.0000	0.0125	55	55.0000	0.0000	0.0126	65	0.0000	65	0.0002	55	55.0000	0.0000	0.0121	55	55.0000	0.0000	0.0122
Caso 5	55	0.0001	55	0.0001	66	66.0000	0.0000	0.0145	55	55.0000	0.0000	0.0146	70	0.0000	70	0.0001	55	55.0000	0.0000	0.0145	55	55.0000	0.0000	0.0146
Caso 6	46	0.0001	46	0.0001	48	48.0000	0.0000	0.0148	46	46.0000	0.0000	0.0149	67	0.0001	67	0.0001	46	46.0000	0.0000	0.0146	46	46.0000	0.0000	0.0146
Caso 7	50	0.0001	50	0.0001	63	63.0000	0.0000	0.0136	50	50.0000	0.0000	0.0137	69	0.0000	69	0.0001	50	50.0000	0.0000	0.0150	50	50.0000	0.0000	0.0151
Caso 8	53	0.0001	53	0.0001	53	53.0000	0.0000	0.0116	53	53.0000	0.0000	0.0117	63	0.0001	63	0.0001	53	53.0000	0.0000	0.0141	53	53.0000	0.0000	0.0142
Caso 9	44	0.0001	44	0.0001	47	47.0000	0.0000	0.0125	44	44.0000	0.0000	0.0125	59	0.0001	44	0.0002	44	44.0000	0.0000	0.0143	44	44.0000	0.0000	0.0144
Caso 10	48	0.0001	48	0.0001	54	54.0000	0.0000	0.0144	48	48.0000	0.0000	0.0145	60	0.0000	60	0.0001	48	48.0000	0.0000	0.0138	48	48.0000	0.0000	0.0139

Scene7

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	23	0.0002	23	0.0003	23	23.0000	0.0000	0.0678	22	22.0000	0.0000	0.0682	23	0.0002	23	0.0003	23	23.0000	0.0000	0.0514	22	22.0000	0.0000	0.0516
Caso 2	22	0.0002	22	0.0003	22	22.0000	0.0000	0.0640	21	21.0000	0.0000	0.0645	23	0.0001	22	0.0002	22	22.0000	0.0000	0.0655	21	21.0000	0.0000	0.0658
Caso 3	24	0.0001	24	0.0002	24	24.0000	0.0000	0.0518	23	23.0000	0.0000	0.0522	24	0.0001	24	0.0002	24	24.0000	0.0000	0.0503	23	23.0000	0.0000	0.0506
Caso 4	26	0.0001	26	0.0003	26	26.0000	0.0000	0.0455	26	26.0000	0.0000	0.0457	26	0.0001	26	0.0002	26	26.0000	0.0000	0.0447	26	26.0000	0.0000	0.0451
Caso 5	24	0.0001	24	0.0002	24	24.0000	0.0000	0.0500	24	24.0000	0.0000	0.0503	24	0.0001	24	0.0002	24	24.0000	0.0000	0.0486	24	24.0000	0.0000	0.0489
Caso 6	27	0.0002	27	0.0002	27	27.0000	0.0000	0.0430	27	27.0000	0.0000	0.0432	27	0.0001	27	0.0001	27	27.0000	0.0000	0.0439	27	27.0000	0.0000	0.0441
Caso 7	25	0.0002	25	0.0003	25	25.0000	0.0000	0.0473	25	25.0000	0.0000	0.0477	25	0.0001	25	0.0002	25	25.0000	0.0000	0.0469	25	25.0000	0.0000	0.0473
Caso 8	24	0.0001	24	0.0002	24	24.0000	0.0000	0.0397	24	24.0000	0.0000	0.0400	24	0.0001	24	0.0001	24	24.0000	0.0000	0.0417	24	24.0000	0.0000	0.0419
Caso 9	22	0.0001	21	0.0002	20	20.4000	0.6633	0.0641	20	20.0000	0.0000	0.0645	28	0.0002	20	0.0003	20	20.4000	0.6633	0.0624	20	20.0000	0.0000	0.0628
Caso 10	26	0.0001	26	0.0002	26	26.0000	0.0000	0.0496	25	25.0000	0.0000	0.0499	26	0.0001	26	0.0001	26	26.0000	0.0000	0.0488	25	25.0000	0.0000	0.0491

Scene8

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	61	0.0001	61	0.0001	73	73.0000	0.0000	0.0472	61	61.0000	0.0000	0.0475	73	0.0001	61	0.0003	61	61.0000	0.0000	0.0394	61	61.0000	0.0000	0.0397
Caso 2	70	0.0001	70	0.0001	83	83.0000	0.0000	0.0478	70	70.0000	0.0000	0.0481	83	0.0001	70	0.0005	70	70.0000	0.0000	0.0452	70	70.0000	0.0000	0.0456
Caso 3	74	0.0001	74	0.0001	87	87.0000	0.0000	0.0507	74	74.0000	0.0000	0.0510	87	0.0001	74	0.0004	74	74.0000	0.0000	0.0518	74	74.0000	0.0000	0.0522
Caso 4	75	0.0001	75	0.0001	90	90.0000	0.0000	0.0380	75	75.0000	0.0000	0.0383	90	0.0001	75	0.0003	75	75.0000	0.0000	0.0373	75	75.0000	0.0000	0.0375
Caso 5	80	0.0001	80	0.0001	95	95.0000	0.0000	0.0501	80	80.0000	0.0000	0.0504	95	0.0001	80	0.0005	80	80.0000	0.0000	0.0527	80	80.0000	0.0000	0.0530
Caso 6	70	0.0001	70	0.0001	81	81.0000	0.0000	0.0351	70	70.0000	0.0000	0.0353	81	0.0001	70	0.0003	70	70.0000	0.0000	0.0337	70	70.0000	0.0000	0.0339
Caso 7	83	0.0001	83	0.0002	96	96.0000	0.0000	0.0479	83	83.0000	0.0000	0.0483	96	0.0000	83	0.0004	83	83.0000	0.0000	0.0514	83	83.0000	0.0000	0.0518
Caso 8	78	0.0001	78	0.0001	89	89.0000	0.0000	0.0353	78	78.0000	0.0000	0.0356	89	0.0001	78	0.0003	78	78.0000	0.0000	0.0383	78	78.0000	0.0000	0.0386
Caso 9	74	0.0001	74	0.0001	86	86.0000	0.0000	0.0497	74	74.0000	0.0000	0.0500	86	0.0000	74	0.0005	74	74.0000	0.0000	0.0489	74	74.0000	0.0000	0.0492
Caso 10	60	0.0001	60	0.0001	75	75.0000	0.0000	0.0442	60	60.0000	0.0000	0.0446	75	0.0001	61	0.0003	60	60.0000	0.0000	0.0488	60	60.0000	0.0000	0.0491

Figura C.2: Escenarios 5 a 8

Figura C.3: Escenarios 9 a 11

Scene9

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	41	0.0002	41	0.0005	41	41.0000	0.0000	0.1282	41	41.0000	0.0000	0.1289	41	0.0002	41	0.0005	41	41.0000	0.0000	0.1245	41	41.0000	0.0000	0.1254
Caso 2	36	0.0002	36	0.0004	36	36.0000	0.0000	0.1480	35	35.0000	0.0000	0.1492	36	0.0001	36	0.0004	36	36.0000	0.0000	0.1579	35	35.0000	0.0000	0.1592
Caso 3	36	0.0001	36	0.0003	36	36.0000	0.0000	0.1508	36	36.0000	0.0000	0.1518	36	0.0002	36	0.0005	36	36.0000	0.0000	0.1506	36	36.0000	0.0000	0.1516
Caso 4	38	0.0001	38	0.0002	38	38.0000	0.0000	0.1231	38	38.0000	0.0000	0.1239	38	0.0002	38	0.0005	38	38.0000	0.0000	0.1218	38	38.0000	0.0000	0.1225
Caso 5	43	0.0001	43	0.0002	43	43.0000	0.0000	0.1160	43	43.0000	0.0000	0.1169	43	0.0002	43	0.0004	43	43.0000	0.0000	0.1140	43	43.0000	0.0000	0.1147
Caso 6	36	0.0001	36	0.0003	36	36.0000	0.0000	0.1567	36	36.0000	0.0000	0.1575	36	0.0002	36	0.0004	36	36.0000	0.0000	0.1557	36	36.0000	0.0000	0.1567
Caso 7	43	0.0002	43	0.0004	43	43.0000	0.0000	0.1230	43	43.0000	0.0000	0.1237	43	0.0001	43	0.0003	43	43.0000	0.0000	0.1256	43	43.0000	0.0000	0.1265
Caso 8	35	0.0001	35	0.0003	35	35.0000	0.0000	0.1737	35	35.0000	0.0000	0.1750	35	0.0002	35	0.0005	35	35.0000	0.0000	0.1758	35	35.0000	0.0000	0.1765
Caso 9	42	0.0002	42	0.0004	42	42.0000	0.0000	0.1867	41	41.0000	0.0000	0.1880	42	0.0003	42	0.0005	42	42.0000	0.0000	0.1884	41	41.0000	0.0000	0.1894
Caso 10	33	0.0003	33	0.0004	33	33.0000	0.0000	0.1891	32	32.0000	0.0000	0.1904	33	0.0002	33	0.0004	33	33.0000	0.0000	0.1935	32	32.1000	0.3000	0.1944

Scene10

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	40	0.0001	40	0.0003	40	40.0000	0.0000	0.0337	39	39.0000	0.0000	0.0339	40	0.0000	40	0.0001	40	40.0000	0.0000	0.0293	39	39.0000	0.0000	0.0294
Caso 2	40	0.0000	40	0.0001	40	40.0000	0.0000	0.0268	40	40.0000	0.0000	0.0270	40	0.0000	40	0.0001	40	40.0000	0.0000	0.0262	40	40.0000	0.0000	0.0263
Caso 3	40	0.0001	40	0.0002	40	40.0000	0.0000	0.0292	39	39.4000	0.4899	0.0294	40	0.0003	40	0.0003	40	40.0000	0.0000	0.0290	39	39.4000	0.4899	0.0292
Caso 4	49	0.0000	49	0.0001	49	49.0000	0.0000	0.0296	49	49.0000	0.0000	0.0298	49	0.0001	49	0.0002	49	49.0000	0.0000	0.0297	49	49.0000	0.0000	0.0298
Caso 5	45	0.0000	45	0.0001	45	45.0000	0.0000	0.0297	45	45.0000	0.0000	0.0298	45	0.0001	45	0.0001	45	45.0000	0.0000	0.0306	44	44.7000	0.4583	0.0307
Caso 6	39	0.0002	39	0.0003	39	39.0000	0.0000	0.0275	39	39.0000	0.0000	0.0276	39	0.0000	39	0.0001	39	39.0000	0.0000	0.0264	39	39.0000	0.0000	0.0266
Caso 7	41	0.0001	41	0.0002	41	41.0000	0.0000	0.0259	40	40.7000	0.4583	0.0261	41	0.0001	41	0.0002	41	41.0000	0.0000	0.0247	40	40.6000	0.4899	0.0248
Caso 8	50	0.0000	50	0.0001	50	50.0000	0.0000	0.0232	50	50.0000	0.0000	0.0233	50	0.0000	50	0.0001	50	50.0000	0.0000	0.0236	50	50.0000	0.0000	0.0237
Caso 9	50	0.0000	50	0.0002	50	50.0000	0.0000	0.0250	49	49.0000	0.0000	0.0252	50	0.0001	50	0.0002	50	50.0000	0.0000	0.0250	49	49.0000	0.0000	0.0252
Caso 10	49	0.0001	49	0.0001	49	49.0000	0.0000	0.0276	48	48.0000	0.0000	0.0278	49	0.0000	49	0.0001	49	49.0000	0.0000	0.0272	48	48.0000	0.0000	0.0274

Scene11

	HCTV		HCTV+FM		HCA				HCA+FM				HCD		HCD+FM		H4				H4+FM			
	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo	Solucion	tiempo	Solucion	tiempo	Solucion	media	desvEst	tiempo	Solucion	media	desvEst	tiempo
Caso 1	23	0.0002	23	0.0018	23	24.1000	0.5385	0.5575	23	23.9000	0.5385	0.5612	23	0.0002	23	0.0029	23	23.0000	0.0000	0.5619	23	23.0000	0.0000	0.5657
Caso 2	25	0.0001	25	0.0019	25	25.9000	0.3000	0.5912	25	25.8000	0.4000	0.5948	25	0.0002	25	0.0014	25	25.0000	0.0000	0.6638	25	25.0000	0.0000	0.6669
Caso 3	24	0.0002	23	0.0033	24	24.3000	0.4583	0.5585	24	24.1000	0.3000	0.5632	24	0.0002	24	0.0020	23	23.9000	0.3000	0.5617	23	23.0000	0.0000	0.5650
Caso 4	26	0.0004	26	0.0022	26	26.8000	0.4000	0.6231	26	26.6000	0.4899	0.6267	26	0.0003	26	0.0018	26	26.0000	0.0000	0.6117	26	26.0000	0.0000	0.6167
Caso 5	26	0.0002	26	0.0027	26	26.8000	0.4000	0.5516	26	26.8000	0.4000	0.5554	27	0.0001	26	0.0037	26	26.0000	0.0000	0.5550	26	26.0000	0.0000	0.5590
Caso 6	23	0.0003	23	0.0045	23	24.0000	0.4472	0.5798	23	24.0000	0.4472	0.5839	23	0.0002	23	0.0026	23	23.0000	0.0000	0.5656	23	23.0000	0.0000	0.5686
Caso 7	26	0.0003	26	0.0031	26	26.9000	0.5385	1.0982	25	26.7000	0.6403	1.1089	26	0.0002	26	0.0019	26	26.0000	0.0000	0.6560	26	26.0000	0.0000	0.6596
Caso 8	25	0.0002	25	0.0022	25	25.0000	0.0000	0.6995	24	24.9000	0.3000	0.7032	25	0.0002	25	0.0019	24	24.8000	0.4000	0.4940	24	24.8000	0.4000	0.4966
Caso 9	26	0.0002	26	0.0015	25	25.6000	0.4899	0.5407	25	25.2000	0.4000	0.5441	24	0.0002	24	0.0015	24	24.0000	0.0000	0.5219	24	24.0000	0.0000	0.5260
Caso 10	25	0.0002	25	0.0029	26	26.9000	0.3000	0.5469	26	26.9000	0.3000	0.5507	26	0.0002	26	0.0026	25	25.0000	0.0000	0.5350	25	25.0000	0.0000	0.5387

Escenario 1

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	16.627	55	11	0	ok	0.3185	11	3.8622	11	11	0	0	ok	Caso 1
Caso 2	17.735	50	10	0	ok	0.1838	10	2.2548	10	10	0	0	ok	Caso 2
Caso 3	34.785	50	9	0	ok	0.0532	9	2.1405	9	9	0	0	ok	Caso 3
Caso 4	9.994	50	10	0	ok	0.0513	10	2.0594	10	10	0	0	ok	Caso 4
Caso 5	102.765	60	11	0	ok	0.0585	11	2.8429	11	11	0	0	ok	Caso 5
Caso 6	17.875	60	12	0	ok	0.0499	12	2.4033	12	12	0	0	ok	Caso 6
Caso 7	50.729	60	10	0	ok	0.0306	10	2.7327	10	10	0	0	ok	Caso 7
Caso 8	41.55	60	9	0	ok	0.0477	9	2.1431	9	9	0	0	ok	Caso 8
Caso 9	34.628	60	10	0	ok	0.0305	10	3.3856	10	10	0	0	ok	Caso 9
Caso 10	12.502	45	8	0	ok	0.0325	8	1.9828	8	8	0	0	ok	Caso 10

Escenario 2

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	15.583	50	5	0	ok	0.1763	5	1.5870	5	5	0	0	ok	Caso 1
Caso 2	13.317	55	6	0	ok	0.1213	6	1.5694	6	6	0	0	ok	Caso 2
Caso 3	14.816	60	5	0	ok	0.0883	5	1.1821	5	5	0	0	ok	Caso 3
Caso 4	24.198	50	5	0	ok	0.1187	5	1.5157	5	5	0	0	ok	Caso 4
Caso 5	9.121	55	5	0	ok	0.1098	5	1.8105	5	5	0	0	ok	Caso 5
Caso 6	11.447	45	5	0	ok	0.1113	5	1.6938	5	5	0	0	ok	Caso 6
Caso 7	12.149	50	5	0	ok	0.0841	5	1.6738	5	5	0	0	ok	Caso 7
Caso 8	8.379	50	5	0	ok	0.1098	5	1.5511	5	5	0	0	ok	Caso 8
Caso 9	20.03	55	5	0	ok	0.1082	5	1.2839	5	5	0	0	ok	Caso 9
Caso 10	8.51	45	5	0	ok	0.0890	5	1.4621	5	5	0	0	ok	Caso 10

Escenario 3

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	10.296	55	11	0	ok	0.1818	11	2.7507	11	11	0	0	ok	Caso 1
Caso 2	14.843	50	10	0	ok	0.0391	10	1.9527	10	10	0	0	ok	Caso 2
Caso 3	16.312	55	11	0	ok	0.0360	11	3.0411	11	11	0	0	ok	Caso 3
Caso 4	14.941	50	10	0	ok	0.0354	10	2.2445	10	10	0	0	ok	Caso 4
Caso 5	13.319	45	8	0	ok	0.0339	8	1.9742	8	8	0	0	ok	Caso 5
Caso 6	25.738	50	10	0	ok	0.0289	10	1.8788	10	10	0	0	ok	Caso 6
Caso 7	16.737	55	9	0	ok	0.0352	9	2.0226	9	9	0	0	ok	Caso 7
Caso 8	17.738	55	11	0	ok	0.0427	11	2.2868	11	11	0	0	ok	Caso 8
Caso 9	20.159	55	11	0	ok	0.0480	11	2.7142	11	11	0	0	ok	Caso 9
Caso 10	30.638	50	10	0	ok	0.0455	10	2.3770	10	10	0	0	ok	Caso 10

Escenario 4

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	254.738	207	17	0	ok	0.0778	17	3.2212	17	17	0	0	ok	Caso 1
Caso 2	227.78	180	15	0	ok	0.0659	15	3.3526	15	15	0	0	ok	Caso 2
Caso 3	775.669	198	16	0	ok	0.0822	16	3.2883	16	16	0	0	ok	Caso 3
Caso 4	151.675	189	13	0	ok	0.0803	13	3.2036	13	13	0	0	ok	Caso 4
Caso 5	225.087	207	17	0	ok	0.0723	17	3.2884	17	17	0	0	ok	Caso 5
Caso 6	515.277	171	13	0	ok	0.0843	13	3.1926	13	13	0	0	ok	Caso 6
Caso 7	83.681	180	15	0	ok	0.1026	15	2.6572	15	15	0	0	ok	Caso 7
Caso 8	171.835	171	14	0	ok	0.0623	14	3.4799	14	14	0	0	ok	Caso 8
Caso 9	70.269	153	13	0	ok	0.0619	13	3.2392	13	13	0	0	ok	Caso 9
Caso 10	108.792	180	14	0	ok	0.0615	14	2.5397	14	14	0	0	ok	Caso 10

Figura C.4: Escenarios 1 a 4

Escenario 5

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	104.87	171	14	0	ok	0.1078	14	2.9951		14	0	0	ok	Caso 1
Caso 2	106.258	207	17	0	ok	0.1008	17	3.8915	17	17	0	0	ok	Caso 2
Caso 3	99.681	180	12	0	ok	0.1147	12	3.7881	12	12	0	0	ok	Caso 3
Caso 4	147.478	171	12	0	ok	0.1115	12	3.0743	12	12	0	0	ok	Caso 4
Caso 5	85.529	189	12	0	ok	0.1100	12	2.9343	12	12	0	0	ok	Caso 5
Caso 6	3605.8	180	15	0.467793	timeout	0.1018	15	3.0884	15	15	0	0	ok	Caso 6
Caso 7	107.093	171	12	0	ok	0.0946	12	2.5201	12	12	0	0	ok	Caso 7
Caso 8	225.66	216	17	0	ok	0.1163	17	6.8133	17	17	0	0	ok	Caso 8
Caso 9	397.117	207	18	0.6944	oom	0.1438	17	5.0420	17	17	0	0	ok	Caso 9
Caso 10	119.818	171	14	0	ok	0.1064	14	3.3521	14	14	0	0	ok	Caso 10

Escenario 6

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	981.693	255	50	0	ok	0.0477	50	12.2169	50	50	0	0	ok	Caso 1
Caso 2	3005.92	235	40	66.66	oom	0.0326	39	67.6385	39	39	0	0	ok	Caso 2
Caso 3	265.284	245	51	36.38	oom	0.0277	48	13.1176	48	48	0	0	ok	Caso 3
Caso 4	443.124	285	281	88.36	oom	0.0373	55	3601.4493	55	*	*	*	timeout	Caso 4
Caso 5	1056.77	280	56	1.79	oom	0.0280	55	3601.3153	55	*	*	*	timeout	Caso 5
Caso 6	3601.44	235	47	0.0212766	timeout	0.0352	46	11.6334	46	46	0	0	ok	Caso 6
Caso 7	311.384	255	54	44.17	oom	0.0300	50	19.2684	50	50	0	0	ok	Caso 7
Caso 8	1810.92	270	54	29.17	oom	0.0372	53	367.5909	53	53	0	0	ok	Caso 8
Caso 9	612.472	230	46	52.80	oom	0.0290	44	339.8735	44	44	0	0	ok	Caso 9
Caso 10	1172.29	245	49	2.04	oom	0.0316	48	3601.4275	48	*	*	*	timeout	Caso 10

Escenario 7

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	943.685	425	24	54.79	oom	0.1439	22	5.1617	22	22	0	0	ok	Caso 1
Caso 2	1312.96	415	22	52.20	oom	0.2533	21	4.9469	21	21	0	0	ok	Caso 2
Caso 3	65.374	485	*	*	oom	0.1448	23	5.0946	23	23	0	0	ok	Caso 3
Caso 4	88.685	470	*	*	oom	0.1315	26	4.5376	26	26	0	0	ok	Caso 4
Caso 5	1405.33	425	*	*	oom	0.1403	24	7.2479	24	24	0	0	ok	Caso 5
Caso 6	162.178	500	*	*	oom	0.1306	27	5.4888	27	27	0	0	ok	Caso 6
Caso 7	946.237	450	*	*	oom	0.1323	25	5.2553	25	25	0	0	ok	Caso 7
Caso 8	1471.6	455	*	*	oom	0.1265	24	7.1879	24	24	0	0	ok	Caso 8
Caso 9	325.016	495	*	*	oom	0.1542	20	5.2279	20	20	0	0	ok	Caso 9
Caso 10	495.704	460	*	*	oom	0.1419	25	5.4300	25	25	0	0	ok	Caso 10

Escenario 8

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR					
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado	
Caso 1	105.808	774	*	*	oom	0.0931	61	10.5541	61	61	0	0	ok	Caso 1
Caso 2	99.252	873	*	*	oom	0.1005	70	12.5546	70	70	0	0	ok	Caso 2
Caso 3	42.819	918	*	*	oom	0.1032	74	12.4568	74	74	0	0	ok	Caso 3
Caso 4	40.425	927	*	*	oom	0.0953	75	1250.8822	75	75	0	0	ok	Caso 4
Caso 5	46.35	999	*	*	oom	0.1089	80	2022.2364	80	80	0	0	ok	Caso 5
Caso 6	33.436	864	*	*	oom	0.0947	70	14.9515	70	70	0	0	ok	Caso 6
Caso 7	35.176	1035	*	*	oom	0.1293	83	1768.8836	83	*	*	*	oom	Caso 7
Caso 8	44.162	963	*	*	oom	0.1055	78	170.2016	78	78	0	0	ok	Caso 8
Caso 9	39.862	918	*	*	oom	0.1090	74	13.8034	74	74	0	0	ok	Caso 9
Caso 10	67.334	828	*	*	oom	0.0986	60	538.8620	60	60	0	0	ok	Caso 10

Figura C.5: Escenarios 5 a 8

Escenario 9

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR				
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado
Caso 1	33.02	999	*	*	com	0.5820	41	9.0067	41	41	0	0	ok
Caso 2	19.298	900	*	*	com	0.4981	36	10.0202	36	35	0	0.03	ok
Caso 3	28.42	882	*	*	com	0.5063	36	11.4007	36	36	0	0	ok
Caso 4	19.569	927	*	*	com	0.6017	38	12.7738	38	38	0	0	ok
Caso 5	30.854	1044	*	*	com	0.5816	43	10.7673	43	42	0	0.02	ok
Caso 6	28.105	918	*	*	com	0.4880	36	3602.1015	36	36	0.0277778	0	timeout
Caso 7	33.619	1053	*	*	com	0.5911	43	9.9027	43	43	0	0	ok
Caso 8	27.039	864	*	*	com	0.5617	35	9.6831	35	35	0	0	ok
Caso 9	31.87	999	*	*	com	0.6592	41	668.7130	41	40	0	0.03	ok
Caso 10	28.792	918	*	*	com	0.6091	32	11.0302	32	32	0	0	ok

Escenario 10

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR				
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado
Caso 1	87.076	615	*	*	com	0.0812	39	1951.1430	39	39	0	0	ok
Caso 2	123.022	555	*	*	com	0.0746	40	3601.3085	40	*	*	*	timeout
Caso 3	82.36	610	*	*	com	0.0825	39	3601.2169	39	*	*	*	timeout
Caso 4	68.496	765	*	*	com	0.0808	49	24.5536	49	49	0	0	ok
Caso 5	78.21	640	*	*	com	0.0894	44	174.4752	44	44	0	0	ok
Caso 6	84.874	635	*	*	com	0.0893	39	18.6609	39	39	0	0	ok
Caso 7	81.07	735	*	*	com	0.0754	40	47.9199	40	40	0	0	ok
Caso 8	105.714	695	*	*	com	0.0777	50	17.1494	50	50	0	0	ok
Caso 9	92.238	640	*	*	com	0.0773	49	20.7880	49	49	0	0	ok
Caso 10	103.408	700	*	*	com	0.0781	48	20.0109	48	48	0	0	ok

Escenario 11

	CPLEX				ESTIMADOR		CPLEX+ESTIMADOR				
	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	Estado	Tiempo (s)	Solución	Tiempo (s)	Período disponible	Solución	CPLEX_GAP	EST_GAP	Estado
Caso 1	114.2	750	750	97.47	com	2.7393	23	72.8782	23	20	0	0.15	ok
Caso 2	423.959	615	615	96.72	com	2.8354	25	91.9614	25	21	0	0.19	ok
Caso 3	86.531	540	540	96.50	com	2.8228	23	1379.3760	23	20	0	0.15	ok
Caso 4	97.544	750	750	97.26	com	2.8674	28	2245.9780	28	22	5.52	0.18	com
Caso 5	389.675	605	605	96.46	com	2.6708	25	901.7015	25	23	6.24	0.09	com
Caso 6	88.109	720	720	97.33	com	2.8780	23	308.2128	23	20	0	0.15	ok
Caso 7	399.76	660	660	96.78	com	2.9306	26	110.9288	26	22	0	0.18	ok
Caso 8	79.13	690	690	97.07	com	2.5417	24	218.8874	24	21	0	0.14	ok
Caso 9	750.969	515	515	96.06	com	2.4032	24	1300.5170	24	21	0	0.14	ok
Caso 10	575.294	635	635	96.73	com	2.6102	25	2655.4450	25	22	4.60	0.14	com

Figura C.6: Escenarios 9 a 11

	Escenario de estrés	
	Solución	Tiempo (seg)
Caso 1, 5 heaters	103666	14,7037
Caso 2, 10 heaters	53073	13,8779
Caso 3, 15 heaters	37154	15,963
Caso 4, 20 heaters	27778	16,0201
Caso 5, 25 heaters	27778	15,4739
Caso 6, 30 heaters	26180	11,5055
Caso 7, 35 heaters	24360	9,0298
Caso 8, 40 heaters	23078	6,5847
Caso 9, 45 heaters	23078	6,1247
Caso 10, 50 heaters	23078	5,7666

Figura C.7: Escenario de estrés

Parámetro	Valor
duración de jornada laboral (min)	480
#heaters	12 (h1...h12)
#moldes	25 (m1...m25)
#moldes disponibles	m1=1, m2=1, m3=2, m4=1, m5=1, m6=2, m7=1, m8=1, m9=1, m10=1, m11=1, m12=1, m13=1, m14=2, m15=1, m16=1, m17=2, m18=2, m19=2, m20=2, m21=1, m22=1, m23=4, m24=1, m25=1
demanda	dm1=45, dm2=35, dm3=600, dm4=35, dm5=85, dm6=271, dm7=185, dm8=155, dm9=215, dm10=315, dm11=115, dm12=45, dm13=18, dm14=515, dm15=54, dm16=55, dm17=413, dm18=621, dm19=673, dm20=227, dm21=35, dm22=18, dm23=825, dm24=25, dm25=55
piezas compartidas	no
tiempo de colocado (min)	m1=50, m2=50, m3=50, m4=50, m5=50, m6=73, m7=50, m8=50, m9=80, m10=80, m11=50, m12=50, m13=73, m14=73, m15=73, m16=50, m17=50, m18=50, m19=80, m20=80, m21=50, m22=73, m23=73, m24=73, m25=80
tiempo de vulcanizado (min)	tv1=32.0, tv2=41.1, tv3=43.2, tv4=41.1, tv5=33.9, tv6=74.4, tv7=28.5, tv8=24.8, tv9=22.3, tv10=19.8, tv11=58.4, tv12=32.0, tv13=80.0, tv14=74.6, tv15=35.6, tv16=17.5, tv17=34.9, tv18=23.2, tv19=15.7, tv20=25.4, tv21=109.7, tv22=106.7, tv23=39.0, tv24=57.6, tv25=26.2
tiempo de quitado (min)	m1=150, m2=150, m3=150, m4=150, m5=150, m6=293, m7=150, m8=150, m9=555, m10=555, m11=150, m12=150, m13=293, m14=293, m15=293, m16=150, m17=150, m18=150, m19=555, m20=555, m21=150, m22=293, m23=293, m24=293, m25=555
compatibilidad molde-molde	$\{m1, m2\}$, $\{m3\}$, $\{m7, m8\}$, $\{m17\}$, $\{m16\}$, $\{m18\}$, $\{m9, m10\}$, $\{m19\}$, $\{m25\}$, $\{m20\}$, $\{m11, m12, m21\}$, $\{m4, m5\}$, $\{m23, m24\}$, $\{m6\}$, $\{m14\}$, $\{m15\}$, $\{m22\}$, $\{m13\}$
compatibilidad molde-heater	$\{h1, h2, h3, h4, h5, h6\} \times \{m1, m2, m3, m7, m8, m9, m10, m16, m17, m18, m19, m20, m25\}$ $\{h7\} \times \{m4, m5, m11, m12, m21\}$, $\{h8, h9, h10\} \times \{m6, m23, m24\}$, $\{h11\} \times \{m14\}$, $\{h12\} \times \{m13, m15, m22\}$,

Tabla C.13: Datos de escenario real