Universidad Tecnológica Nacional
Facultad Regional Santa Fe


# Modelos Avanzados de Optimización para la Gestión Eficiente de Procesos de Producción

*Doctorado en Ingeniería Mención Industrial*

Doctoranda: Ing. Natalia Paola Basán

Director: Dr. Carlos Alberto Méndez

*Diciembre de 2018*

# *Agradecimientos*

*El desarrollo de esta tesis doctoral fue posible gracias al acompañamiento académico y personal de familiares, compañeros y amigos.*

*Agradezco en primer lugar a mi director de tesis, el Dr. Carlos Méndez, por su dedicación permanente, sus concejos y su apoyo constante a lo largo de mi carrera profesional. Por su gran confianza hacia mi trabajo, sus aportes y por promover constantemente nuevas líneas de investigación y espacios de debate.*

*Muy especialmente a la Dra. Mariana Cóccola, por su valiosa dedicación, esfuerzo y aportes, los cuáles hicieron posible la concreción de varios trabajos académicos y capítulos de esta tesis.*

*A mis padres, Patricia y Mario, por alentarme y darme la posibilidad de formarme profesionalmente, por enseñarme con el ejemplo a ser una mejor persona y superarme día a día, por transmitirme sus valores y por la confianza y apoyo constante e incondicional en cada momento de mi vida. A mi querido hermano, Gabriel, mi confidente y amigo, por su infinita paciencia, por su cariño y contención. A mi compañero de vida, Mariano, por su comprensión y apoyo constante, por su amor y confianza a lo largo de estos últimos años de camino juntos.*

*A mis compañeros de oficina, Victoria y Pedro, y a todas las personas que conocí durante el desarrollo de este trabajo, por las discusiones diarias, las reuniones amenas y fructíferas, por su amistad y los más gratos momento de trabajo y recreación.*

*A todos los que integran el grupo CAPSE, Vanina, Agustín, Diego, Pablo, Luis, Nélida, Rodolfo y Jaime, por darme la posibilidad de formar parte de un grupo de trabajo y de contención tan ameno, enriqueciéndome con sus experiencias y consejos.*

*A la Universidad Tecnológica Nacional, muy especialmente a los directivos y docente, por brindarme todo lo necesario para mi formación doctoral.*

*Y finalmente, al Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), por el soporte económico ofrecido, y al Instituto de Desarrollo Tecnológico para la Industria Química (INTEC) y la Facultad de Ingeniería Química de la Universidad Nacional del Litoral, por ofrecerme el ámbito de trabajo más adecuado para llevar a cabo mis labores de investigación y desarrollo.*

# *Contenido*

# RESUMEN

En la actualidad, la industria se desenvuelve en mercados globales con una creciente competitividad, donde, independientemente de su tamaño y del sector de actividad involucrada, debe conciliar la satisfacción de sus clientes con la alta exposición a presiones, exigencias de calidad, márgenes de rentabilidad cada vez más estrechos y una elevada eficiencia económica. Estas tendencias han determinado que la gestión de soporte de las actividades de manufactura y distribución de bienes industriales se transforme en un área clave dentro de las organizaciones. Las compañías necesitan utilizar los recursos disponibles de una manera eficiente y mejorar continuamente sus procesos para tener una respuesta más rápida y efectiva ante las posibles variaciones del mercado. Las empresas no sólo deben administrar sus procesos internos, sino que también deben disponer de mejores soluciones logísticas para lograr importantes ventajas competitivas a través de (i) una reducción de los costos, (ii) una mayor capacidad de reacción ante los cambios permanentes de la demanda, (iii) una mejor utilización de los recursos, (iv) un incremento de la productividad, (v) un mayor grado de satisfacción del cliente y, no menos importante, (vi) una reducción en el impacto ambiental.

En particular, en el contexto industrial moderno, las decisiones que hacen a la gestión y operación de una empresa deben ir acompañas de herramientas computacionales cada vez más eficientes, de manera de auxiliar a los procesos de toma de decisiones. Por esta razón, en las últimas décadas, los esfuerzos por perfeccionar las técnicas de modelado de procesos industriales se han focalizado en el desarrollo de

metodologías eficientes para dar solución a los problemas de programación de operaciones ("scheduling") de mediano y corto plazo.

A lo largo de los capítulos de la tesis se propone el análisis y desarrollo de herramientas computacionales, basadas en técnicas de optimización de tipo riguroso y heurístico, para dar soporte al problema de programación de operaciones en diferentes tipos de procesos productivos, los cuáles se caracterizan por ser complejos y de gran escala. En la primera parte de la tesis, se presentan y analizan diferentes enfoques alternativos de optimización exacta para encontrar el mejor programa de producción para una industria que se dedica a la construcción de grandes barcos. Dicho proceso productivo agrega una complejidad adicional al considerar operaciones de ensamblado. Las metodologías propuestas están basadas en modelos de Programación Matemática Lineal Mixta-Entera ("Mixed-Integer Linear Programming", MILP) con dominio de tiempo continuo. El principal objetivo de la compañía es minimizar el tiempo total requerido para producir y ensamblar todas las partes que integran cada barco.

A continuación, los modelos matemáticos utilizados para la resolución del problema de programación de operaciones en la industria naval, son tomados como base para el desarrollo de un algoritmo de descomposición que tiene como principal objetivo proporcionar soluciones "rápidas" y de buena calidad a problemas complejos de programación de operaciones de escala industrial en sistemas de manufacturas de tipo flow shop flexibles. Se lleva a cabo un análisis comparativo entre ambos tipos de metodologías de solución propuestas: (i) basadas en métodos rigurosos de optimización, y (ii) basadas en técnicas de descomposición.

Posteriormente, se integran las decisiones de programación de las operaciones de piso de planta de entornos industriales de tipo job shop flexibles, con decisiones de rediseño, a través de una estrategia de solución basada en formulaciones matemáticas MILP. De este modo, no sólo se determina el mejor programa de operaciones, sino que se evalúan reconfiguraciones factibles que permiten aumentar la eficiencia del uso de los recursos de sistemas de manufactura flexible.

La potencialidad de las metodologías presentadas se evalúa a través de la resolución de diferentes instancias de casos de estudio reales de gran escala vinculados a la industria naval, que sirven de base, tanto para evaluar la eficiencia computacional de las formulaciones propuestas y la calidad de las soluciones halladas, como también para realizar un análisis comparativo de las contribuciones de este trabajo de tesis.

Por último, se presenta una formulación matemática MILP basada en una representación discreta del tiempo, para modelar y optimizar las decisiones operativas en procesos que presentan un consumo intensivo de energía eléctrica y que operan en mercados diarios de precios de energía sensibles a los períodos de tiempo, es decir, que varían hora a hora. El modelo matemático propuesto tiene como principal objetivo determinar el programa de operaciones ("schedule") óptimo de manera predictiva, para un horizonte de planificación determinado, estableciendo los modos de operación y los niveles de producción de plantas dedicadas a la separación de aire, de manera de satisfacer la demanda y al mismo tiempo, minimizar el costo energético total. Es importante mencionar que, el enfoque de optimización presentado está basado en una novedosa representación del proceso de transición de estados de operación, involucrados en este tipo de plantas, lo que permite obtener una herramienta capaz de considerar eficientemente las fluctuaciones de los precios del mercado y al mismo tiempo, optimizar las decisiones operativas correspondientes a los procesos de consumo intensivo de energía, requiriendo un mínimo esfuerzo computacional. Este modelo es aplicado a un caso de estudio real de escala industrial con el fin de mostrar su alta eficiencia y robustez.

Asimismo, cabe destacar que, diversos modelos de simulación y metodologías híbridas, se han desarrollado como trabajo de investigación, en el marco de esta tesis doctoral, para hacer frente al problema de "scheduling" de casos de estudios reales. En particular, se han propuesto diferentes enfoques de simulación de eventos discretos como herramientas para la toma de decisiones de procesos industriales, las cuales permiten determinar "schedules" eficientes y evaluar posibles inversiones, nuevas políticas de operación, diferentes estrategias de reingeniería de procesos, entre otros. Sin embargo,

por razones de espacio y organización, no se presentan en detalle en esta tesis. Dichas contribuciones se pueden encontrar en diferentes reportes científicos que se detallan más adelante.

Las nuevas herramientas desarrolladas, junto con los resultados obtenidos, a lo largo de este trabajo de investigación, han sido presentadas en diversos congresos internacionales referentes en el área de ingeniería de procesos y sistemas, y divulgados a través de publicaciones en revistas científicas nacionales e internacionales de alto impacto y en capítulos de libros. A continuación, se describen los trabajos publicados y el medio de difusión.

## Publicaciones en revistas científicas internacionales y nacionales de divulgación

- Básan, N., Cóccola, M.E., Garcia-del-Valle A. and Méndez, C.A. **2018**. *"Integrated Design and Scheduling of Flexible Manufacturing Plants: A MILP-based Decomposition Algorithm and Case Studies"*. **Computers & Industrial Engineering Journal**. ISSN: 0360-8352 (enviado)

- Básan, N., Cóccola, M.E., Garcia-del-Valle A. and Méndez, C.A. **2018**. *"An Efficient MILP-Based Decomposition Strategy for Solving Large-Scale Scheduling Problems in the Offshore Oil and Gas Industry"*. **Optimization and Engineering Journal**. ISSN: 1573-2924 (enviado)

- Básan, N., Achkar, V.G., Garcia-del-Valle A. and Méndez, C.A. **2018**. "*An Effective Continuous-Time Formulation for Scheduling Optimization in a Shipbuilding Assembly Process*". **Iberoamerican Journal of Industrial Engineering**. ISSN: 2175-8018. (aceptado, en edición)

- Basán, N.P., Grossmann, I.E., Gopalakrishnan, A., Lotero, I., and Méndez, C.A. **2018**. "*Novel MILP Scheduling Model for Power-Intensive Processes under Time-Sensitive*

*Electricity Prices*". **Industrial and Engineering Chemistry Research**, vol. 57, pp. 1581-1592. ISSN: 08885885.

- Basán, N.P., Pulido, R., Cóccola, M.E., and Méndez, C.A. **2017**. "*Aerospace manufacturing industry: a simulation-based decision support framework for the scheduling of complex hoist lines*". **Iberoamerican Journal of Industrial Engineering**. vol. 9, n. 17. ISSN: 2175-8018.

- Básan, N., Cóccola, M.E., and Méndez, C.A. **2014**. *"An innovative discret event simulation tool to improve the efficiency of a complex beer packaging line".* **Anales de la Academia Nacional de Ciencias Exactas, Físicas y Naturales de Buenos Aires**, tomo 65, pp. 61-74. ISSN: 0365-1185.

## Capítulos de libros y anales en congresos con referato

- Basán N. P., Achkar V. G., Garcia-del-Valle A. and Méndez C.A. **2017**. *"A Heuristic Simulation-Based Framework to Improve the Scheduling of Blocks Assembly and the Production Process in Shipbuilding".* **2017 Winter Simulation Conference**, pp. 3218-3229. ISBN: 978-153863428-8.

- Basán, N.P., Grossmann, I.E., Gopalakrishnan, A., Lotero, I. and Méndez, C.A. **2017**. *"Optimal scheduling for power-intensive processes under time-sensitive electricity prices".* **Computer Aided Chemical Engineering**, vol. 40, pp. 1423-1438. ISSN: 15707946.

- Basán N. P., Achkar V. G., Garcia-del-Valle A. and Méndez C.A. **2017**. *"A Hybrid Simulation-Based Optimization Approach for Scheduling Dinamic Block Assembly in Shipbuilding".* **29th European Modeling & Simulation Symposium**, EMSS 2017, pp. 83-90. ISBN: 978-151084765-1.

- Basán N.P. and Méndez C.A. **2016.** *"Hybrid MILP/Simulation/Heuristic Algorithms to Complex Hoist Scheduling Problems".* **Computer Aided Chemical Engineering,** vol. 38, pp. 1929-1934. ISBN: 978-044463428-3.

- Basán N.P., Cóccola M.E. and Méndez C.A. **2015.** *"Conducting Experimental Design and Optimization of the System Configuration and Operation of an Innovative Car Rental Bussiness".* **27th European Modeling & Simulation Symposium**, EMSS 2015, pp. 166-171. ISBN: 978-889799948-5.

- Básan, N., Cóccola, M., and Méndez, C.A. **2014**. "*Optimizing the design and operation of a beer packaging line through and advanced SIMIO-based DES tool*", **2014 Winter Simulation Conference**, pp. 1989-2000. ISBN: 978-1-4799-7486-3.

- Basán N.P., Ramos, L., Cóccola M.E., and Méndez C.A. **2013**. *"Modeling, simulation and optimization of the main line packaging of a brewing company".* **25th European Modeling & Simulation Symposium**, EMSS 2013, pp. 551-560. ISBN: 978-889799922-5.

## Presentaciones en congresos internacionales y nacionales

- Basán N. P., Garcia-del-Valle A., Faulin J. and Méndez C. **2017**. "*Effective continuous-time formulations for scheduling shipyard block assembly system*". **23th Conference of the International Federation of Operational Research Societies – IFORS 2017**. Quebec, Canada.

- Basán N. P., Achkar V. G., Garcia-del-Valle A. and Méndez C. **2017**. *"An effective continuous-time formulation for scheduling optimization in a shipbuilding assembly process".* **46 Jornadas Argentinas de Informática – 46 JAIIO**. Córdoba, Argentina.

- Basán, M., Cóccola, M., and Méndez, C. **2016**. *"Production scheduling optimization for power-intensive processes with time-sensitive electricity prices".* **45 Jornadas Argentinas de Informática – 45 JAIIO**. Buenos Aires, Argentina.

- Basán N.P., Méndez C.A., Grossmann I., Gopalakrishnan A. and Lotero I. **2016**. *"Towards Optimal Production of Industrial Gases with Uncertain Energy Prices".* **Aiche Annual Meeting - 16 AIChE**. San Francisco, EE.UU.

- Basán, N. P., Cóccola M and Méndez C. **2015**. *"Aerospace manufacturing industry: a simulation-based decision support framework for the scheduling of complex hoist lines".* **44 Jornadas Argentinas de Informática – 44 JAIIO**. Rosario, Argentina.

- Basán N. P., Aguirre A., and Méndez C. **2014.** *"Advanced optimization and simulation-based tool for complex automated manufacturing systems".* **20th Conference of the International Federation of Operational Research Societies – IFORS 2014**. Barcelona, España.

# *Lista de figuras*

# *Lista de tablas*

# Capítulo 1

## Introducción y fundamentos teóricos

### 1.1. Introducción

La creciente competitividad y los fenómenos de globalización a los cuales se enfrentan las compañías industriales en la actualidad, les exige respuestas cada vez más eficientes, y procesos y estrategias que les permitan sobrevivir y crecer en un mundo en continuo cambio, en el cual el cliente es quien asume, cada vez más, el poder de negociación y quien al final define el éxito o fracaso de todo el engranaje industrial que se encuentra tras la fabricación de un producto. Hoy en día, las empresas se enfrentan a un mapa tecnológico de gran complejidad, en el cual se integran cada uno de los procesos de la misma, desde el eslabón más primordial de la producción hasta el proceso de distribución. En este contexto, el continuo avance tecnológico, reglas ambientales más estrictas, productos con ciclos de vida cada vez más cortos, fuertes limitaciones de recursos y márgenes de rentabilidad más estrechos obligan a las empresas productoras de

bien y servicios a buscar mejores alternativas para la ejecución de sus procesos. Al mismo tiempo, dadas las condiciones actuales de competencia global, la utilización eficiente de recursos limitados y el cumplimiento de los compromisos pactados con los clientes traen inmediatamente aparejado un incremento en la productividad y rentabilidad de la empresa. Estos aspectos resultan ser indispensables para las empresas de manufactura innovadoras que quieran ganar participación en el mercado, operar a su máxima eficiencia y exceder las expectativas del cliente.

En muchos mercados, la habilidad de las empresas para entregar sus productos a mayor velocidad, les permite obtener ventaja respecto a los competidores con características, calidad y precio de producto similares. En otros mercados, la entrega rápida puede justificar un sobreprecio y ciertamente, mejorar el grado de satisfacción del cliente. En todos los casos, los tiempos de entrega más cortos incrementan la flexibilidad y agilidad de la planta, reducen la necesidad de inventarios de seguridad y disminuyen el riesgo de obsolescencia. Para alcanzar este objetivo es esencial utilizar enfoques innovadores que beneficien conjuntamente a todos los actores involucrados.

De esta manera, el actual entorno globalizado y altamente competitivo impone que hoy en día las industrias se vean enfrentadas principalmente a tres grandes retos: (i) la optimización de recursos y procesos, (ii) la reducción de los costos y (iii) la disminución de los riesgos. Sin embargo, llegar a tener una planta con las características mencionadas es complejo. Los sistemas de producción reales están sujetos a un número creciente de situaciones controlables y otras restricciones imposibles de controlar desde el interior de la organización. Entre estas situaciones se pueden nombrar las diferencias de tiempo de procesamiento entre un proceso y otro, la variedad de artículos a procesar en una misma máquina, la cancelación inesperada de pedidos, las detenciones no programadas de las máquinas, la entrada de pedidos urgentes, la eficiencia de las personas, los problemas de calidad, la disponibilidad de los materiales, entre otros.

Todas estas contingencias repercuten inevitablemente en las actividades del corto plazo, que en ocasiones retrasan los programas de producción planteados. Esto ocasiona a

menudo cambios en el proceso normal, improvisaciones y en ocasiones, detenciones en los procesos, repercutiendo en la planificación establecida por parte de los administradores y programadores de la producción.

Frente a este panorama, las tareas de gestión y operación de procesos industriales deben abarcar la administración de inventarios, manejo de materiales, compras, programación de procesos de manufactura, transporte, administración de almacenes y centros de distribución, control de estándares de servicio al cliente, entre otros. En particular, el proceso de programación de las operaciones de producción cumple un papel preponderante en la industria, en la medida que busca reducir costos y lograr altos niveles de productividad para que las empresas mejoren su competitividad. Es necesario, por lo tanto, que dichas empresas innoven no sólo en sus procesos productivos sino también en sus herramientas de gestión.

De este modo, se pone de manifiesto la necesidad de contar con herramientas sistemáticas de soporte al proceso de toma de decisiones en los niveles estratégico, táctico y operacional. Tal es así que, distintas técnicas y procedimientos sistemáticos, han sido tratados e implementados para abordar problemas de programación de operaciones de corto, mediano y largo plazo en ambientes industriales. Algunas de ellas, como las nuevas técnicas de simulación (modelos descriptivos), permiten alcanzar en gran medida los objetivos a través de la experimentación y análisis de escenarios visualizados, donde se puede conocer el comportamiento de las variables en el tiempo, realizar modificaciones experimentales de los parámetros del sistema y conocer las estadísticas e indicadores para tomar decisiones basadas en información exacta y oportuna. Por otro lado, la programación matemática y las técnicas heurísticas (modelos predictivos), han surgido para optimizar rendimientos, parámetros o especificaciones del sistema. Mediante la consideración y análisis conjunto de las alternativas viables, estas herramientas apuntan a hallar la configuración más adecuada en virtud de un objetivo preciso.

Cabe destacar que, entre los principales beneficios que trae aparejado contar con herramientas eficientes para la programación detallada de las operaciones, se pueden

mencionar: (i) garantizar que se logre una utilización óptima de la capacidad de producción, lo que reduce el tiempo de inactividad y el exceso de uso de los equipos; (ii) garantizar que el nivel de inventario se mantenga en niveles óptimos en todo momento, es decir, que no haya exceso o faltante de existencias y (iii) garantizar que el tiempo de producción se mantenga en un nivel óptimo, aumentando la productividad.

A continuación, se hace una breve introducción de los problemas de programación de las operaciones en ambientes industriales, puntualizando las principales características y clasificación de los sistemas productivos en los que se basa el desarrollo de las diferentes metodologías presentadas en este trabajo de tesis. Seguidamente, se efectúa una revisión general de los trabajos existentes en esta área, para luego, exponer el objetivo principal de la tesis. Por último, se detallan tanto los diversos tipos problemas industriales de programación de operaciones abordados a lo largo de este trabajo, como las metodologías de solución propuestas para afrontarlos.

## 1.2. Programación detallada de las operaciones en ambientes industriales

El desarrollo de nuevas e innovadoras herramientas computacionales de soporte, tanto para la *Planificación de las Operaciones* a mediano y largo plazo, como para la función de control de operaciones en piso de planta, llamada *Programación de las Operaciones* de corto plazo de plantas industriales, también conocida por su término en inglés "*scheduling*", han sido temas de especial interés para abordar los problemas que debe afrontar el sector industrial moderno. La diferencia entre planificación y programación está en el nivel de detalle y horizonte temporal de las decisiones involucradas en cada una. La planificación se determina para el mediano y largo plazo con un nivel agregado de detalle, mientras que la programación se define para el corto plazo con un nivel detallado. Por otra parte, el proceso de planificación de la producción sitúa las necesidades en intervalos temporales suponiendo capacidad infinita o, a lo sumo, realizando verificaciones simplificadas de capacidad. Al contrario, la programación de la producción

genera las órdenes de fabricación habiendo verificado la disponibilidad de materiales, de recursos y de mano de obra directa.

Para optimizar la toma de decisiones de las compañías industriales se debe considerar un proceso jerárquico formado por tres niveles con diferentes escalas de tiempo (Castro et al., 2018), cada uno de los cuales persigue objetivos específicos y utiliza diferentes herramientas y metodologías (Baldea y Harjunkoski, 2014; Brunaud y Grossmann, 2017; Maravelias y Sung, 2009): (i) diseño, donde se toman decisiones en cuanto a la estructura, diseño, rediseño, reingeniería, entre otros; (ii) planificación, donde se tienen en cuenta las decisiones correspondientes a alcanzar las metas de producción y (iii) programación de las operaciones o "scheduling", último eslabón, donde se toman las decisiones correspondientes a la especificación de las operaciones de producción, estimación de la duración, asignación de recursos, etc.; y se ejecutan los mecanismos para su control. Los desafíos relacionados a la integración de estos niveles de decisión son reconocidos en el área de "Enterprise-wide Optimization" (EWO) (Grossmann, 2005, 2012). El objetivo de EWO es abordar la planificación, scheduling, optimización en tiempo real y control de inventarios en sistemas de procesos industriales de forma integrada. La Figura 1.1 muestra los tres niveles mencionados, con sus respectivas escalas de tiempo.

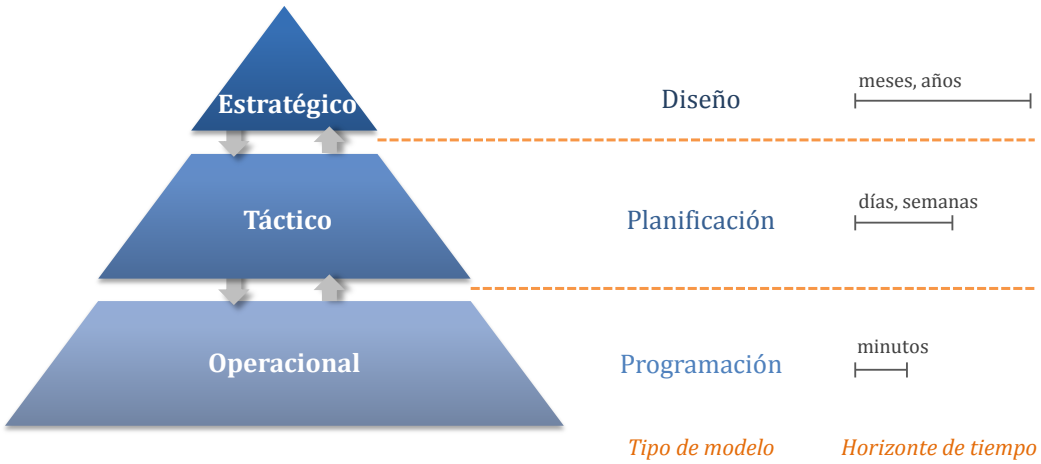



Figura 1.1. Diseño, planificación y programación.

En particular, este trabajo de tesis aborda la optimización de la programación de las operaciones de manufactura presentes en diferentes tipos de sistemas productivos. Es importante considerar que los problemas de programación de operaciones poseen tres características principales:

1. *El entorno de manufactura*: tiene que ver con el flujo y las instalaciones de producción.

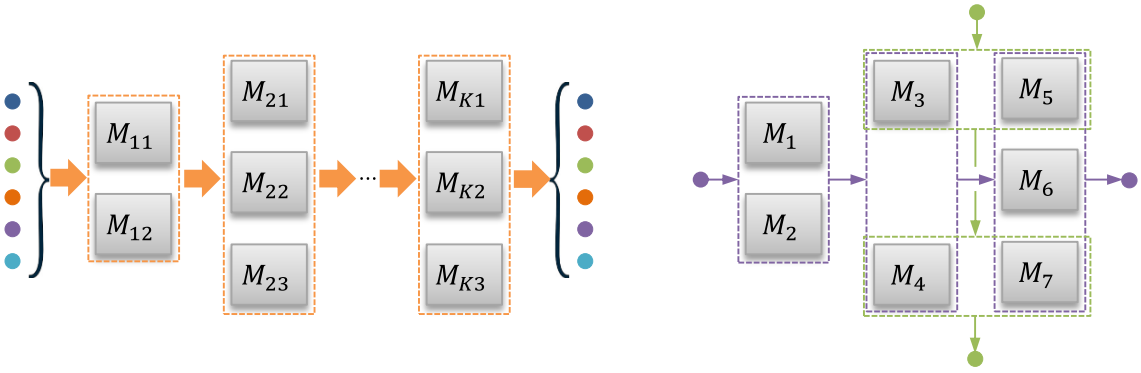
2. *Las restricciones*: normalmente, los entornos de manufactura se encuentran limitados por ciertas características, que pueden estar relacionadas con la secuencia de procesamiento, restricciones tecnológicas, de capacidad, de requerimiento de recursos, temporales, entre otras.

3. *Una función objetivo*: existen varias medidas de rendimiento usadas para describir los problemas de programación de operaciones, tales como el tiempo de terminación, la demora o tardanza de un trabajo, el costo total involucrado, entre otros.

En cuanto al entorno de manufactura, en la presente tesis, se consideran los dos principales tipos de procesos de fabricación en los que se pueden clasificar los problemas de programación, que según la literatura, de acuerdo a las instalaciones, el número de etapas de procesamiento, las limitaciones operativas y la complejidad de la ruta de producción, pueden ser: (i) *flow shop* y (ii) *job shop*, además de su respectivas variantes (Taylor y Pan, 2007). Según la teoría de la complejidad, ambos tipos de problemas son considerados "NP-hard" (Garey et al., 1976), es decir, que los requerimientos computacionales para obtener una solución óptima crecen exponencialmente a medida que lo hace el tamaño del problema. La clase "NP" es el conjunto de todos los problemas de decisión que pueden ser resueltos por algoritmos no deterministas en tiempo polinómico.

El *problema de programación de las operaciones de producción en sistemas flow shop* ("Flow Shop Scheduling Problem" - FSP) se define como un conjunto de productos, tareas o trabajos que se procesan secuencialmente en múltiples máquinas en el mismo

orden, es decir, todos tienen la misma receta de procesamiento y el flujo de productos en la planta es unidireccional. Debido al interés de algunas empresas de ampliar sus capacidades productivas, resulta necesario aumentar la flexibilidad en la producción mediante el uso de líneas paralelas o unidades de procesamiento paralelas. De este modo, las etapas de procesamiento resultan de la integración de unidades o equipos que operan en paralelo para llevar a cabo la elaboración de los productos. Esta variante del FSP se conoce como el *problema de programación de las operaciones en sistemas flow shop flexibles* ("Flexible Flow Shop Scheduling Problem" - FFSP) y comúnmente surge en entornos extensivos de producción secuencial de múltiples etapas, como se muestra en la Figura 1.2. Lee y Loong (2019) señalan que el FFSP se puede clasificar en función del número de etapas, trabajos y máquinas o unidades de procesamiento. Además, según Zandieh y Gholami (2009), las unidades de una misma etapa que realizan operaciones similares, pueden clasificarse como idénticas, uniformes o no relacionadas. En general, se consideran aquellas que tienen la misma capacidad y operan a la misma velocidad, es decir, las definidas como "idénticas".




(a) Flow shop flexible (b) Job shop flexible

Figura 1.2. Entornos productivos: multietapa y multipropósito.

A diferencia del FSP, en el problema de programación de operaciones de producción de sistemas de manufactura tipo *job shop* ("Job Shop Scheduling Problem" - JSP), los trabajos pueden seguir secuencias de procesamiento diferentes y no necesariamente empleando todas las etapas del proceso productivo (Benttaleb et al.,

2018; Corsano et al., 2007; Tamssaouet et al., 2018). Para extender su aplicabilidad a los sistemas de fabricación flexibles, donde las máquinas o unidades de procesamiento se caracterizan por tener una alta flexibilidad (Rossi, 2014), el clásico problema JSP se generaliza para definir al *problema de programación de operaciones de sistemas job shop flexibles* ("Flexible Job Shop Scheduling Problem" - FJSP). La definición de FJSP establece que se puede realizar una operación de manufactura en un conjunto de unidades de producción consideradas compatibles (ver Figura 1.2). Estos sistemas de manufactura también son conocidos como "multipropósito", ya que se componen de unidades de procesamiento que pueden realizar las operaciones correspondientes a más de una etapa productiva del proceso. De acuerdo con Shen et al. (2018), la disponibilidad de unidades alternativas destinadas a realizar operaciones similares puede aumentar el rendimiento de la planta y al mismo tiempo, ayudar a gestionar el mantenimiento, tanto preventivo como corregir averías y otros eventos imprevistos. Tanto el JSP, como el FJSP, son problemas clásicos, que debido a su presencia en el mundo real, principalmente en los sectores industriales y de servicios, ha despertado un importante interés entre los investigadores.

Por otro lado, se debe tener en cuenta que al considerar múltiples etapas, las plantas pueden operar con diferentes políticas de espera, almacenamiento intermedio y transferencia de productos entre dichas etapas de procesamiento. Las políticas que se consideran comúnmente en la literatura son (Majozi et al., 2017; Rippin, 1996):

- *Almacenamiento intermedio ilimitado* ("Unlimited Intermediate Storage" - UIS): sin restricciones de almacenamiento, suponiendo que existe capacidad suficiente para contener todo el material generado en las diferentes etapas de procesamiento.

- *Almacenamiento intermedio finito* ("Finite Intermediate Storage" - FIS): se considera que existe capacidad limitada para almacenar el material generado en cada etapa.

- *Sin almacenamiento intermedio* ("Non Intermediate Storage" - NIS): esta política se basa en que no existen sectores de almacenamiento intermedio entre las etapas productivas. Cuando finaliza el procesamiento de un lote o una orden de trabajo, esta debe permanecer en la unidad donde fue procesada hasta que una máquina de la siguiente etapa se encuentre disponible.

- *Tiempo de espera ilimitado* ("Unlimited Wait" - UW): el material puede aguardar en el equipo que lo procesó, por tiempo indeterminado, hasta que la siguiente etapa de procesamiento se encuentre disponible.

- *Tiempo de espera limitado* ("Finite-Wait" - FW): esta política se utiliza cuando el material que ha sido procesado en una etapa debe ser transferido al poco tiempo a la etapa siguiente.

- *Transferencia sin espera* ("Zero Wait" - ZW): una vez finalizado el procesamiento de un producto en una etapa, es inmediatamente transferido a la etapa siguiente, no admitiendo esperas. Por lo tanto, no existen almacenamiento intermedio, dando lugar a una política NIS/ZW, y se adopta por ejemplo, cuando la calidad de un producto puede deteriorarse al interrumpir el procesamiento del mismo.

Independientemente del entorno productivo, el objetivo del problema de "scheduling" es encontrar un programa detallado de operaciones de piso de planta, o también conocido como "schedule" de corto plazo, orientado a incrementar, tanto la rentabilidad y productividad, como el nivel de servicio brindado a los clientes, determinando qué, cuándo y cuánto se debe fabricar. Las decisiones a considerar son:

a. el equipo o unidad de procesamiento donde se debe procesar cada tarea, lote u orden de trabajo, según corresponda;

b. los tiempos de inicio y finalización del procesamiento de cada una de las órdenes de trabajo involucradas;

c. la secuencia de procesamiento de las órdenes en cada unidad disponible;

con el objetivo de satisfacer todas las restricciones operativas y, en lo posible, las fechas de entrega de dichas órdenes, optimizando, al mismo tiempo, algún criterio de medida de la calidad del "schedule" generado.

Es importante destacar que, entre las restricciones más relevantes a considerar, se encuentran las derivadas de la disponibilidad limitada de recursos (equipos, materia prima, mano de obra, servicios, entre otros), de precedencia, recetas de procesamiento secuenciales (lineales) o ramificadas (no-lineales) que impliquen división o mezcla de lotes de productos, prioridades de las órdenes a atender en los diferentes períodos, capacidad, tiempos de transición dependientes de la secuencia de procesamiento, "release-times" de equipos y materias primas, fechas de entrega, turnos de trabajo, longitud del horizonte de programación, períodos de mantenimiento, política de transferencia y almacenamiento de inventario, y recursos de transporte y movimiento interno de materiales. Asimismo, las restricciones del problema pueden clasificarse en dos tipos: las que deben satisfacerse necesariamente (factibilidad), y las que pueden relajarse, o alcanzar un cierto grado de cumplimiento (calidad).

Por último, como se ha mencionado anteriormente, para identificar el mejor programa de operaciones de la planta, es necesario contar con una medida de calidad, usando criterios de rentabilidad, productividad y/o nivel de satisfacción del cliente. Dicho criterio de evaluación es la función objetivo del problema cuyo valor se desea optimizar. Las más estudiadas son las siguientes:

- ✓ *Tiempo total de producción o "makespan"*: es el tiempo total requerido para completar el procesamiento de todas las órdenes de trabajo, y representa un objetivo atractivo para gran parte de las compañías industriales, ya que persigue la utilización eficiente de los recursos.

- ✓ *Anticipación y tardanza total en la entrega* ("Lateness total"): considera la suma de los retrasos de los trabajos con respecto a su fecha de entrega

("due date"), ya que se evalúa la conformidad de los plazos establecidos. Particularmente, en este caso, los costos de inventario están asociados a la finalización temprana de un requerimiento, es decir, la anticipación con respecto a su fecha de entrega, y los costos de insatisfacción de los clientes se vinculan a la falta de cumplimiento de las fechas de entrega previamente pactadas.

- ✓ *Costo de operación:* se reducen los costos de manera que se maximice la eficiencia del sistema productivo y con esto, los beneficios.

## 1.3. Revisión literaria

Acorde a la definición introducida en la sección anterior, el proceso de "scheduling" de corto plazo de plantas industriales, en términos generales, busca definir la mejor manera de asignar los recursos disponibles para atender los requerimientos de demanda del mercado y cumplir con todas las restricciones operativas, de tal modo de satisfacer los objetivos de las compañías (Castro et al., 2018; Pinedo, 2016). En las últimas dos décadas, las técnicas de optimización basadas en métodos exactos de programación matemática han jugado un papel importante para resolver dicho problema, teniendo en cuenta decisiones tales como la asignación de tareas a unidades de equipamiento y la secuenciación de dichas tareas en cada unidad (Harjunkoski et al., 2014; Li y Ierapetritou, 2008; Maravelias, 2012; Méndez et al., 2006; Pinto y Grossmann, 1998; Maravelias y Sung, 2009), donde el principal objetivo es encontrar la solución óptima al problema de programación de operaciones de producción con un esfuerzo computacional razonable. En este sentido, existen importantes contribuciones en el área, desarrolladas para resolver exitosamente problemas académicos, es decir, de tamaño moderado, a través de formulaciones basadas en Programación Matemática Lineal Mixta-Entera ("Mixed-Integer Linear Programming", MILP). Las propuestas más relevantes combinan formulaciones de tiempo continuo basadas en períodos de tiempos (también llamados "time-slots") y restricciones de precedencia (Castro et al., 2012; Pinto y Grossmann, 1995; Castro y

Grossmann, 2006, 2005), como así también basadas en conceptos de precedencia, tanto inmediata (Cerdá et al., 1997; Méndez et al., 2000), como general global (Méndez y Cerdá, 2002; Méndez et al., 2001; Carlos Alberto Méndez y Cerdá, 2003).

Siguiendo esta dirección, se pueden encontrar otros trabajos reportados en la literatura, principalmente basados en modelos de programación matemática, tales como los que incluyen metodologías heurísticas (Pinedo, 2008; Ruiz y Maroto, 2005) y meta-heurísticas (França et al., 1996; Osman Y Kelly, 1996), programación por restricciones (Harjunkoski y Grossmann, 2002; Jain y Grossmann, 2001; Maravelias y Grossmann, 2004), herramientas de simulación de eventos discretos (Banks et al., 2010; Law, 2007; Wainer, 2009) y técnicas de descomposición (Ierapetritou y Floudas, 1998; Roslöf et al., 2002, 2001, 1999). Si bien estas metodologías se encuentran sustentadas en fuertes desarrollos teóricos, las mismas fueron mayormente utilizadas para la resolución de problemas de tipo académico, es decir, de pequeña y mediana escala (Méndez et al., 2006), debido a su elevado costo computacional y sus limitaciones para la representación de diversos aspectos de los problemas que se intentan resolver.

En los ambientes industriales actuales el problema de "scheduling" implica programar el procesamiento de cientos de productos o lotes de productos en una gran cantidad de unidades o máquinas disponibles para cada tarea, de acuerdo a largas secuencias de etapas de procesamiento, donde además, se pueden considerar diferentes políticas de transferencia y almacenamiento intermedio. Estas decisiones están altamente interconectadas haciendo difícil la resolución de los modelos completos para obtener y asegurar la optimalidad global de las soluciones reportadas, imposibilitando su aplicación a problemas de gran tamaño. En estos casos, el rendimiento computacional de los modelos de optimización resulta ser inestable e impredecible. Como resultado, el uso de métodos exactos como único enfoque de resolución no resulta conveniente para este tipo de problemas.

En este contexto, la situación actual expuesta motiva fuertemente, tanto a nivel académico como industrial, el desarrollo de nuevas estrategias alternativas de solución

que resulten apropiadas para las problemáticas reales del sector industrial moderno. Así, se pueden encontrar investigaciones centradas en el desarrollo de técnicas heurísticas o metaheurísticas para la resolución eficiente de los problemas de programación de operaciones en entornos industriales (Pan y Ruiz, 2013; Ruiz y Stützle, 2008), las cuales se basan en reglas de despacho, algoritmos genéticos, teoría de grafos, "tabu-search", métodos de optimización de colonias de hormigas, entre otros.

Otro aspecto central a considerar a la hora de abordar problemas reales, es el tratamiento de las condiciones de incertidumbre, inherentes a la variabilidad del proceso y a la dinámica propia del sector de manufactura industrial. En este sentido, las metodologías de solución basadas en modelos de simulación de eventos discretos, permiten obtener una representación estadística apropiada de las condiciones de incertidumbre en demandas, precios, materias primas, productos, equipamiento, etc., a partir de la utilización del concepto de interacción de procesos con un conjunto de objetos predefinidos, destinados a representar las características principales de dichos sistemas (Banks et al., 2010). La disponibilidad de lenguajes de simulación de alto nivel orientados a problemas específicos, el uso de herramientas automáticas de análisis estadístico de datos y resultados y la posibilidad de recurrir a modos de visualización en dos y tres dimensiones, han convertido a la simulación en una de las herramientas más populares para el análisis, diseño y evaluación de sistemas, destacándose particularmente aquellos relacionados a la manufactura y/o distribución de bienes industriales. Diversos trabajos de investigación, que han sido publicados en revistas y congresos internacionales de primer nivel, resaltan el creciente interés de la industria y la academia en este tipo de herramientas de computación (Aguirre et al., 2008; Basán et al., 2017e, 2017b, 2015, 2014; Longo, 2013; Wolfsmayr et al., 2015).

Por otra parte, en la revisión de la literatura académica, pueden encontrarse trabajos que se focalizan en la utilización de métodos híbridos (Chu et al., 2014; L. Zhang et al., 2016). En particular, para la resolución de problemas MILP de gran tamaño, la clave del éxito es encontrar formulaciones que permitan reducir el espacio de búsqueda de las

soluciones factibles del problema, con el objetivo de garantizar un comportamiento de los modelos más estable y predecible. Por esta razón, el desarrollo de enfoques híbridos basados en simulación y optimización resulta una estrategia de solución muy conveniente para obtener resultados de buena calidad en procesos de gran escala (Basán et al., 2017b; Castro et al., 2011; Klemmt et al., 2009; Kulkarni y Venkateswaran, 2015). Asimismo, se pueden encontrar contribuciones científicas basadas en diferentes técnicas de descomposición-agregación y algoritmos de mejora para el problema de programación de operaciones en sistemas de manufactura de múltiples etapas (Aguirre et al., 2012; Cóccola et al., 2015; Harjunkoski y Bauer, 2017; Kopanos et al., 2010). Los métodos de descomposición se integran con las formulaciones matemáticas con el fin de reducir el número de variables binarias involucradas, con respecto a los enfoques exactos de espacio completo, reduciendo el espacio de búsqueda y con esto, el esfuerzo computacional requerido para resolver el problema de "scheduling" de escala industrial.

Cabe destacar que, si bien estrategias de solución mencionadas no garantizan las optimalidad de las soluciones reportadas, desde el punto de vista industrial, permiten obtener soluciones de buena calidad en un tiempo computacional razonable.

Como resultado de todo lo expresado, los potenciales beneficios de los sistemas híbridos de soporte al proceso de toma de decisión aparecen como un área emergente de conocimiento en constante expansión, especialmente en el campo de la Ingeniería Industrial aplicada a los procesos de manufactura y distribución. Por esta razón, y con el propósito de afrontar los desafíos planteados, en este trabajo de tesis se aportan diferentes enfoques y metodologías de solución basadas en el desarrollo de modelos de programación matemática y procedimientos heurísticos, para la optimización de decisiones a corto plazo, y en otros a mediano plazo, donde se deben tomar decisiones sobre la configuración de los sistemas productivos.

## 1.4. Objetivos de la tesis

El objetivo principal de este trabajo de tesis consiste en el desarrollo de herramientas sistemáticas basadas en metodologías integradas de reglas heurísticas y métodos de optimización, para la resolución de problemas de programación de operaciones y toma de decisión durante la gestión y operación de procesos complejos de interés industrial, a fin de superar limitaciones de trabajos previos en dicha área.

Los objetivos específicos de esta tesis son los siguientes:

- Formular métodos rigurosos alternativos de optimización, de manera de evaluar las soluciones encontradas para problemas de "scheduling" de los procesos industriales de escala real.

- Desarrollar estrategias de solución a través de procedimientos iterativos y sistemáticos, basados en metodologías de descomposición y mejora de la solución, para problemas industriales complejos de sistemas de manufactura de tipo flow shop flexibles que requieren, tanto operaciones de producción, como de ensamblado de productos intermedios.

- Analizar y comparar la aplicabilidad, eficiencia y performance computacional de los enfoques exactos, basadas en programación lineal mixta-entera (MILP), y los procedimientos iterativos de descomposición-agregación formulados, mediante casos de estudio reales.

- Desarrollar metodologías de descomposición-agregación, basadas en programación matemática, para considerar decisiones de rediseño y programación de las operaciones de plantas de manufactura flexibles, y analizar el vínculo entre ambas decisiones para distintos escenarios.

- Formular modelos de optimización, basados en programación matemática y en una representación sistemática de procesos, que consideran esquemas de estados de transición y operación posibles en la planta.

- Aplicar las diferentes estrategias de solución desarrolladas a procesos industriales, utilizando datos reales.

Por lo tanto, sobre la base de lo expresado en las secciones previas y la definición del alcance de la tesis, en este trabajo se realizan contribuciones en diversas áreas de investigación, tales como enfoques de optimización basados en formulaciones matemáticas, técnicas heurísticas y estrategias sistemáticas, híbridas e integradas basadas en la programación matemática y en técnicas de descomposición-agregación, dando soporte al proceso de toma de decisión de corto y mediano plazo. Asimismo, se realiza un análisis de dichas contribuciones con el fin de llevar a cabo una evaluación objetiva en cuanto a: (i) eficiencia de las herramientas desarrolladas, (ii) optimalidad de las soluciones, (iii) robustez de los modelos y (iv) adaptabilidad de las estrategias de solución.

Teniendo en cuenta este último punto, los modelos y estrategias de solución son evaluados a partir de su aplicación en casos de estudio reales de diversos sectores de la industria, tales como la industria naval y aquellas que tienen como principal materia prima la energía eléctrica, entre las que se encuentra la industria destinada a la separación de aire. De este modo, se trabaja con diferentes tipos y configuraciones de plantas de manufactura: de un producto, multiproducto, multietapa y multipropósito, siguiendo la misma receta, con recetas heterogéneas, con operaciones de ensamblado y con estados de operación y transición.

Para concretar las actividades mencionadas anteriormente, se utiliza principalmente, el sistema de modelado algebraico genérico GAMS (“General Algebraic Modeling System”) 24.9.2 junto con el código de optimización lineal mixto entero CPLEX 12.7.1.0 y el optimizador GUROBI 6.5.2. Generalmente, el criterio de terminación utilizado para la resolución de los modelos es de una diferencia de optimalidad o GAP relativo (“optimality GAP”) del 0%, asegurando la optimalidad global de los mismos, ó 3600 segundos de tiempo de CPU.

## 1.5. Estructura general – Síntesis de los problemas abordados

Como se ha mencionado en la sección anterior, en este trabajo de tesis se propone el abordaje de distintos problemas vinculados a la programación de operaciones de procesos de manufactura complejos de diferentes sectores de la industria actual. A continuación, se presenta la organización del contenido en lo que resta de la tesis, que concretamente está organizada en seis capítulos, en cada uno de los cuales se detalla los problemas abordados, las metodologías desarrolladas para resolverlos, los casos de estudio reales donde han sido aplicadas y una discusión detallada de los resultados obtenidos.

En el **Capítulo 2** se aborda un problema de programación de operaciones de procesamiento y ensamblado de la industria naval, específicamente del proceso de construcción de grandes buques. El sistema bajo estudio está compuesto por múltiples etapas de procesamiento con unidades idénticas en paralelo, capaces de procesar una gran cantidad de productos. Con el objetivo de determinar el programa óptimo de operaciones de manufactura de este tipo de problemas, que minimice el tiempo total requerido para el montaje de una embarcación de gran escala, se proponen tres formulaciones matemáticas MILP alternativas con representación continua del tiempo: (i) basada en intervalos de tiempo ("time-slots"), (ii) basada en precedencia inmediata y (iii) basada en el concepto de precedencia general global. Estos modelos son aplicados a diferentes instancias de un caso de estudio real de un astillero europeo, donde se considera la producción de un gran número de sub-bloques y bloques, y el ensamblado riguroso de los mismos según una secuencia predefinida en función de la estrategia de construcción del buque en cuestión. El objetivo principal es finalizar el proyecto (montaje de una embarcación) en el menor tiempo posible y de acuerdo con los recursos disponibles. Las soluciones reportadas permiten llevar a cabo una evaluación y análisis comparativo de las capacidades computacionales de los enfoques rigurosos de optimización para problemas de diferentes tamaños: desde la programación de pocas órdenes de productos, hasta tamaños reales de astilleros, donde se debe coordinar una gran variedad y cantidad de recursos para

programar todas las órdenes involucradas. Los resultados demuestran la alta complejidad combinatoria que caracteriza a este tipo de problemas de "scheduling", la cual excede fácilmente las capacidades de resolución actualmente proporcionadas por los softwares de optimización disponibles para problemas de tamaño mediano y grande. Las formulaciones MILP presentadas en este capítulo y los correspondientes resultados obtenidos han sido publicados en los anales de los congresos *46 Jornadas Argentinas de Informática – 46 JAIIO* (2017) y *23th Conference of the International Federation of Operational Research Societies – IFORS 2017,* como así también, próximos a ser publicados en la revista *Iberoamerican Journal of Industrial Engineering* (2018). Es importante mencionar que, el modelo matemático basado en el concepto de "time-slots" ha sido utilizado para el desarrollo de una metodología hibrida, junto con un modelo de simulación de eventos discretos. Tanto el procedimiento iterativo que integra ambos modelos, como los resultados reportados, se pueden encontrar en un capítulo de libro de *29th European Modeling & Simulation Symposiu, pp. 83-90* (*EMSS 2017*).

Con el fin de aprovechar la robustez de los métodos exactos abordados en el Capítulo 2, y desarrollar una estrategia de solución eficiente para problemas de programación de operaciones de tamaños industriales, en el **Capítulo 3** se presenta una metodología iterativa que tiene como base la formulación MILP desarrollada, la cual está basada en el concepto de precedencia general global. De este modo, el algoritmo iterativo propuesto integra los métodos de optimización y las técnicas heurísticas, con el fin de explotar los principales beneficios de ambos enfoques para la resolución de problemas complejos de programación: por un lado, la robustez y la rigurosidad de los modelos matemáticos y por otro, la flexibilidad de los procedimientos de mejora y de descomposición-agregación. De este modo, el procedimiento permite generar y mejorar gradualmente la solución del problema de "scheduling" de forma iterativa, secuenciando múltiples tareas a la vez. El algoritmo consta de dos etapas principales: (i) una primera *fase de construcción* de una solución global inicial y factible, y (ii) una segunda *fase de mejora* gradual y sistemática de dicha solución. El objetivo principal es establecer el programa óptimo, o cercano al óptimo, de las operaciones de producción y ensamblado de

un sistema flow shop flexible (FFSP) de gran tamaño en un tiempo computacionalmente aceptable. La estrategia de solución presentada se aplica a la resolución del caso de estudio abordado en el Capítulo 2, y a otro caso, correspondiente también a un sistema productivo flow shop, pero sobre la configuración de otro astillero. Los resultados computacionales demuestran que el algoritmo de descomposición propuesto es capaz de obtener, en todos los casos analizados, soluciones de alta calidad en pocos segundos de tiempo de CPU. En otras palabras, reduce de manera significativa los requerimientos computacionales para resolver el problema de "scheduling", frente a los métodos exactos anteriormente propuesto. Estos resultados han sido oportunamente enviados para su publicación a la revista internacional *Optimization and Engineering Journal* (2018).

En el **Capítulo 4**, se propone una metodología rigurosa alternativa basada en un modelo matemático MILP para establecer el mejor plan detallado de operaciones y evaluar el rediseño de una planta de manufactura de tipo job shop. Por lo tanto, se propone abordar el problema FJSP de sistemas de manufactura flexibles, lo cual implica el procesamiento de una variedad de productos con diferentes secuencias de procesamiento, y además, al igual que los capítulos anteriores, procesos que pueden requerir operaciones de ensamblado de productos intermedios. Asimismo, se considera una segunda etapa del problema FJSP, donde se evalúa el rediseño de la configuración del sistema, teniendo en cuenta las restricciones propias del proceso bajo estudio, tales como restricciones físicas, operativa, entre otras. De este modo, se analizan: (i) posibles cuellos de botellas, (ii) utilización de las unidades de cada estación de trabajo, (iii) operaciones que puede realizar cada equipo y (iv) posibles cambios de configuración para mejorar la productividad del sistema. Por lo tanto, se propone una estrategia de descomposición de naturaleza híbrida heurística-algorítmica para dar solución al problema de "scheduling" planteado, con el objetivo de minimizar el makespan y luego, dar solución al problema de rediseño, de manera de mejorar la eficiencia del proceso productivo, y por consiguiente, el valor de la solución de programación de las operaciones. Este procedimiento iterativo finaliza cuando la solución hallada no puede ser mejorada a través del conjunto de cambios permitidos. Mediante la aplicación de esta estrategia se obtuvieron resultados

altamente satisfactorios, los cuales se encuentran reflejados en los casos de estudio analizados en dicho capítulo, inherentes a la industria automotriz y naval. El material relacionado con el problema abordado en este capítulo ha sido enviado a la revista *Computers & Industrial Engineering Journal* (2018) para su correspondiente publicación.

Cabe mencionar que, en los capítulos detallados se trabaja con datos reales, a los cuales se ha podido acceder a través de un proyecto de cooperación entre la compañía naval y la Universidad da Coruña (Ferrol, España).

En el **Capítulo 5** se propone una formulación de programación matemática MILP de corto plazo con representación discreta del tiempo para el problema de programación de operaciones diarias, inherentes a procesos que presentan un consumo intensivo de energía eléctrica. Debido a que, generalmente, estos procesos involucran un conjunto de modos de operación específicos de grandes equipos, donde existen fuertes interacciones energéticas, cambios relativamente pequeños en sus variables de decisión pueden dar lugar a ahorros considerables de energía, y por lo tanto, de los costos asociados. El objetivo es obtener un programa detallado de operaciones óptimo, de manera predictiva, de las actividades de producción, determinando los modos de operación y los niveles de producción que satisfacen la demanda, a fin de minimizar el costo total de energía durante un horizonte de tiempo determinado. Cabe destacar que, esta nueva formulación está basada en un novedoso concepto, llamado "*Process State Transition Network*" (PSTN), desarrollado para modelar sistemáticamente las posibles secuencias de procesamiento y transiciones entre los diferentes estados de operación de una planta. El modelo involucra variabilidad en el tiempo, tanto para la demanda, como para los precios de la energía eléctrica, y diversas restricciones operativas, tales como modos de transición factibles, tiempos mínimos de residencia, niveles de inventario permitidos, entre otros. El mismo es testeado con datos reales de una reconocida compañía internacional, que posee plantas de gran escala dedicadas a la separación de aire y que operan en mercados diarios de compra-venta de energía. En particular, se determina el "schedule" óptimo teniendo en cuenta precios de la energía sensibles a los períodos de tiempo, para un horizonte de

programación de la producción de una semana. Los resultados reportados demuestran que el modelo propuesto permite modelar y optimizar las decisiones operativas de dicho proceso eficientemente, requiriendo un esfuerzo computacional de pocos segundos de tiempo de CPU. Las principales contribuciones asociadas a este capítulo pueden encontrarse en los anales del congreso *Aiche Annual Meeting - 16 AIChE* (2016) y en un capítulo de libro publicado en *Computer-Aided Chemical Engineering*, vol. 40, pp. 1423-1438 (2017). Una versión extendida de los anteriores, se ha publicado en la revista *Industrial & Engineering Chemistry Research,* vol. 57, pp. 1581-1592 (2018).

Finalmente, el **Capítulo 6** resume las contribuciones más importantes del trabajo de tesis así como un listado de trabajos pendientes que se pretenden abordar a futuro.

# Capítulo 2

## *Programación de operaciones en sistemas de manufactura de la Industria naval: Modelos MILPs alternativos*

## 2.1. Introducción

En el marco de la gestión de la producción, la programación de las operaciones a corto plazo es una etapa clave en las decisiones de carácter operativo, donde se busca el cumplimiento de las metas propuestas mediante la asignación eficiente de los recursos disponibles a las diferentes tareas de producción. Por esta razón, la programación y el control de la ejecución de las operaciones involucradas en un proceso de manufactura constituyen dos actividades que han adquirido una importancia relevante en las últimas décadas, en especial en los procesos de fabricación a demanda, es decir, que no son en serie o por lotes. Uno de los campos de mayor aplicación es la industria naval, debido a que se halla en permanente evolución, ya sea por la continuidad con la que ocurren las innovaciones tecnológicas, como por la propia mutabilidad del proceso productivo. La

internacionalización de este tipo de industrias las ha obligado a invertir en el desarrollo de tecnologías y métodos cada vez más competitivos.

La construcción naval es un proceso de producción complejo de largo plazo que involucra la fabricación de diferentes tipos de embarcaciones para el transporte y comercio marítimo, como petroleros, graneleros, transportistas de carga, etc., y de diversas instalaciones para la explotación de recursos marinos. Por lo tanto, representa un proceso productivo que requiere la coordinación de una gran cantidad de recursos limitados durante un período de tiempo prolongado. Además, la fabricación de embarcaciones e instalaciones marinas tiende a ser a gran escala y muy poco estandarizada, por lo que el proceso de manufactura requiere tanto, una considerable cantidad de mano de obra, como tecnología moderna, impactando directamente en los costos de la compañía (Ciccantell y Shin, 2009).

Tradicionalmente, el proceso de construcción naval se llevó a cabo a través de un enfoque orientado a proyectos. Sin embargo, debido a que los barcos generalmente se construyen con pocos componentes, los cuales se basan en el mismo diseño pero cada uno con un cierto grado de personalización, comenzó a implementarse hace varias décadas, un enfoque modular basado en procesos de estandarización y principios de ajustes. Este enfoque consiste en el uso de una *Estrategia de Construcción* basada en un *diseño modular integrado*, donde se lleva a cabo la prefabricación de grandes estructuras de acero, también llamadas *bloques*, que luego se ensamblan en una grada o en un dique seco para conformar la embarcación. A su vez, cada bloque puede resultar del ensamble de dos o más *sub-bloques*, componentes de menor tamaño, siendo la embarcación el producto final resultante de la agregación de todos los productos intermedios en los que ha sido dividida previamente. El diseño de cada una determina la cantidad y el tamaño de cada bloque y sub-bloque a ser procesado. La Figura 2.1 muestra un ejemplo de una estrategia de construcción, donde se usa un diseño modular para la construcción de un barco: el casco se divide en bloques y cada bloque, a su vez, se divide en sub-bloques.

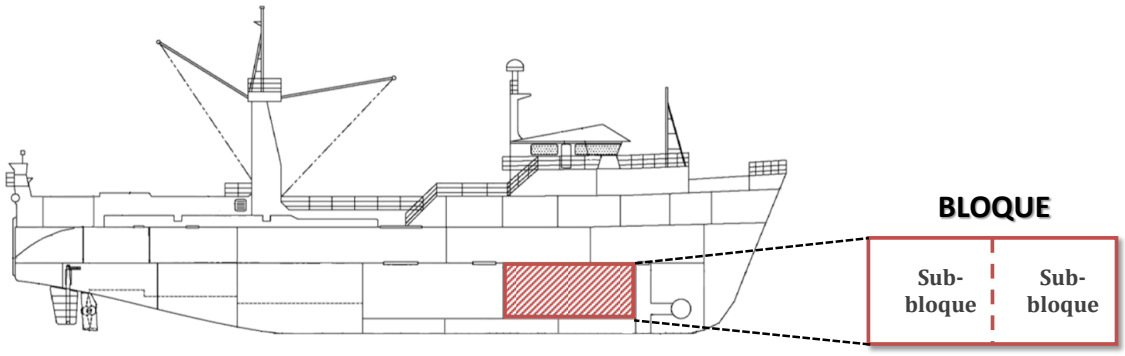



Figura 2.1. Estrategia de división en bloques y sub-bloques - Enfoque de construcción modular integrado.

El proceso de fabricación de embarcaciones, o también llamado *proceso de ensamblado de bloques,* es considerado un cuello de botella para las industrias navales. Esto se debe a que dicho proceso requiere de un alto grado de coordinación entre los diversos recursos disponibles para satisfacer, tanto las restricciones de producción, como las políticas de almacenamiento y requerimientos operativos, implicando una alta complejidad en el sistema de producción. Un retraso en la finalización del proyecto (embarcación) aumenta el riesgo de incurrir en sanciones en la entrega. Además, la inactividad de cualquier recurso, como por ejemplo un dique seco, representa una importante pérdida de ingresos para la compañía.

Dada la complejidad de los procesos involucrados, como la cantidad de componentes (bloques y sub-bloques) a producir, los recursos necesarios, las diferentes políticas de almacenamiento, la demanda, etc., se vuelve indispensable contar con un sistema de programación de operaciones que pueda reportar planes de producción de máxima eficiencia, donde se requieran un mínimo de horas-hombre (Cho et al., 1998). A pesar de que en las últimas décadas han surgido desarrollos significativos en técnicas de modelado, pocos trabajos se han focalizado en la programación de este tipo de proceso. El principal desafío para este tipo de problemas es generar programas eficientes de las operaciones de producción, es decir, que requieran un esfuerzo computacional moderado,

teniendo en cuenta todas las tareas y aspectos productivos involucrados. Particularmente en el proceso de ensamblado de bloques de la industria naval, como se ha mencionado, se debe considerar la gran cantidad de bloques y sub-bloques que deben ser procesados y ensamblados para construir una determinada embarcación.

Se pueden encontrar en la literatura diversos trabajos que abordan el proceso de construcción de barcos. Entre las diferentes propuestas de metodologías de resolución se encuentran métodos heurísticos, metaheurísticos y basados en simulación, con el objetivo de analizar las decisiones operativas y proporcionar soluciones viables con tiempos computacionales razonables. Algunos de estas contribuciones presentan algoritmos heurísticos para determinar la programación de las operaciones, mejorando el uso del área de trabajo a largo plazo y minimizando los tiempos de procesamiento de los bloques (Koh et al., 2008; Shang et al., 2017). Otros autores, como Lee et al. (2009), Liu et al. (2011) y Cebral-Fernandez et al. (2016), se basaron en técnicas de modelado y simulación de eventos discretos para desarrollar herramientas que permitan obtener un programa de operaciones y de uso de los recursos disponibles, que maximice la productividad y reduzca los tiempos involucrados. Basán et al. (2017) también propusieron un ambiente basado en simulación de eventos discretos aplicando diferentes reglas heurísticas, con el fin de evaluar el flujo de materiales y minimizar el tiempo total requerido para la construcción de barcos. Este enfoque resultó de gran utilidad para representar un sistema a nivel industrial y analizar su comportamiento dinámico a lo largo del tiempo, logrando de este modo un buen programa inicial de las tareas productivas llevadas a cabo en un astillero. Sin embargo, estas investigaciones reportadas en la literatura se basan en criterios heurísticos simples, donde ninguno de ellos puede asegurar una solución óptima al problema de programación planteado.

Desde la perspectiva operacional, el tipo de configuración productiva del proceso de ensamblado de bloques puede considerarse como un sistema *flow shop flexible*, ya que se debe procesar un conjunto de órdenes de productos a lo largo de una serie de etapas consecutivas siguiendo la misma secuencia de procesamiento, en donde algunas de las

etapas de producción tienen unidades idénticas en paralelo. Asimismo, se debe considerar que los productos no son procesados en todas las etapas productivas.

Como se ha mencionado en el Capítulo 1, el problema de programación de operaciones en una sistema flow shop flexible *("Flexible Flow Shop Scheduling Problem" - FFSP)* ha sido objeto de estudio de una gran cantidad de trabajos de investigación y publicaciones (Castro et al., 2010; Chen et al., 2017; Cheng et al., 2018; Choi y Wang, 2012; Kopanos et al., 2010). Este tipo de problemas consiste en determinar el "schedule" de un sistema de manufactura compuesto de varias etapas productivas, donde pueden variar los tiempos de procesamiento y se pueden considerar diferentes restricciones de almacenamiento intermedio, con el fin de optimizar el proceso en términos de cierta función objetivo.

Respecto a las soluciones planteadas para este tipo de problemas también se han reportado métodos híbridos. Por ejemplo, en la investigación realizada por Xiong et al. (2015), los autores propusieron una estrategia de solución híbrida, donde presentaron un modelo MIP ("Mixed Integer Programming"), para resolver un problema de programación de un sistema flow shop con operaciones de ensamblado, incorporando heurísticas y metaheurísticas para tratar instancias de tamaños del problema medianos y grandes. Esta formulación resulta útil para el modelado del proceso de construcción naval ya que, como se mencionó anteriormente, se trata de un problema de programación de un sistema de producción y ensamblado de tipo flow shop.

Entre los enfoques determinísticos existentes para resolver el problema de programación FFSP, se encuentran las formulaciones matemáticas MILP (Castro et al., 2009; Kopanos et al., 2009; C.A. Méndez et al., 2001; Pinto y Grossmann, 1995). El principal objetivo de estos métodos exactos de optimización es encontrar la solución óptima para la programación de la producción con un esfuerzo computacional razonable. Las decisiones involucradas se refieren a la asignación de las tareas a unidades de procesamiento y la secuenciación de dichas tareas en una misma unidad (Méndez et al.,

2006; Maravelias y Sung, 2009). Siguiendo esta línea, en este capítulo, se proponen tres formulaciones matemáticas basadas en *programación matemática lineal mixta-entera* (MILP) con representación continua del tiempo para el problema de programación de operaciones del proceso de ensamblado de bloques. El objetivo es generar un programa óptimo de operaciones que permita minimizar el tiempo total requerido para el montaje de una embarcación de tamaño industrial, conocido como criterio del makespan. Las formulaciones introducidas incorporan restricciones para representar las operaciones de ensamblado. Los modelos matemáticos abordados son los siguientes:

- Modelo MILP basado en intervalos de tiempo, comúnmente llamados "*time-slots*" (Basán et al., 2017; Pinto y Grossmann, 1995).
- Modelo MILP basado en *precedencia inmediata* (Cerdá et al., 1997).
- Modelo MILP basado en *precedencia general global* (Méndez et al., 2001).

La resolución del problema radica en determinar: (i) la asignación de las tareas a las unidades de procesamiento de cada etapa del proceso de construcción naval, (ii) la secuencia de producción y ensamblado de los diferentes tipos de productos intermedios (bloques y sub-bloques) en cada etapa y, (iii) los tiempos iniciales y finales de procesamiento de cada uno de ellos. Teniendo en cuenta que existe una gran cantidad de órdenes de trabajo y que las etapas están formadas por unidades individuales e idénticas en paralelo, la complejidad combinatoria del problema se ve incrementada. Se utiliza un conjunto de instancias de diferentes tamaños adaptadas de un caso de estudio real para analizar y evaluar el desempeño de cada modelo.

El capítulo está organizado de la siguiente manera. En la sección 2.2 se introduce la definición del problema, describiendo el proceso clásico de ensamblado de bloques en un astillero y además, exponiendo las principales hipótesis a ser modeladas. Luego, en la sección 2.3, se describe la metodología de solución propuesta, describiendo las formulaciones MILP utilizadas y las adaptaciones para representar el problema bajo

estudio. Seguidamente, en la sección 2.4, se muestran los resultados computacionales obtenidos de los enfoques alternativos de optimización, y finalmente, en la sección 2.7 se describen las conclusiones alcanzadas.

## 2.2. Descripción del problema

### 2.2.1. Proceso de construcción naval

En el proceso de construcción naval, el primer paso es determinar el diseño de la embarcación que se desea construir, y con esto identificar sus diferentes componentes: bloques y sub-bloques. Un bloque es la unidad de construcción más grande de una embarcación. Asimismo, una embarcación se puede dividir en decenas de bloques de tamaño específico, según su diseño (Figura 2.1). A su vez, cada bloque puede surgir del ensamble de dos o más sub-bloques, donde estos últimos se componen de placas de acero. Ambos se consideran productos intermedios en el diseño modular de una embarcación.

En las etapas iniciales del procesamiento, se construyen los sub-bloques a partir de placas y perfiles de acero, para luego, ser montados y formar los grandes bloques. En las etapas siguientes, estos bloques son procesados secuencialmente, para finalmente ser trasladados a un dique seco y ensamblados de manera de constituir el casco del barco, operación conocida como *Montaje final* o *Montaje en grada.* Generalmente, los bloques llegan al dique con el todo equipamiento necesario instalado, sin embargo, puede ocurrir que durante la operación de montaje se deban colocar elementos adicionales, tales como tuberías, soportes y equipos electrónicos.

En la Figura 2.2 se pueden observar las etapas que conforman un proceso típico de construcción de barcos. Dicho proceso comienza con la etapa de *Prefabricación,* que tiene como materias primas chapas y perfiles de acero, los cuales se cortan en piezas pequeñas y se sueldan para formar las unidades más pequeñas de una embarcación (sub-bloques). Luego, el proceso continúa con la operación *Prearmamento 1 – Primera parte*, que consiste

en la instalación de diferentes componentes, tales como tuberías, soportes y elementos auxiliares. El producto obtenido de esta etapa se considera que son los sub-bloques terminados.

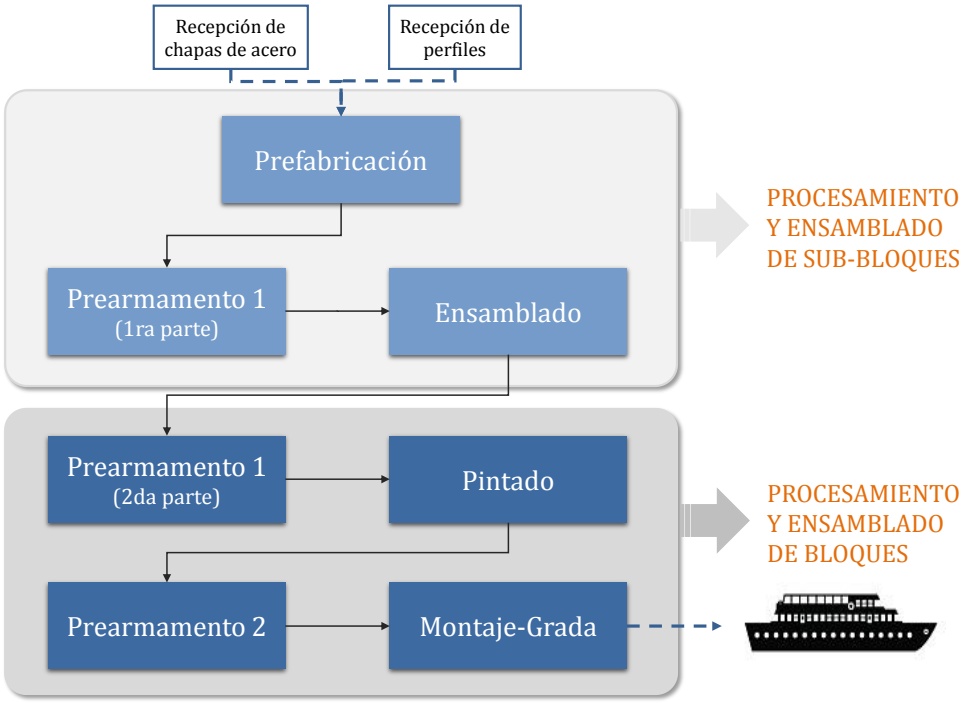



Figura 2.2. Etapas del proceso de construcción naval.

Una vez que los sub-bloques están equipados, son ensamblados a través de operaciones de soldadura para formar la estructura de los bloques (operación conocida como de *Unión* o *Ensamblado*). Las siguientes etapas del proceso de fabricación se refieren a la terminación y el transporte de los bloques a la grada. Una vez ensamblado cada bloque, se realiza una segunda operación de equipamiento de elementos en la estructura de cada uno de ellos, proceso llamado *Prearmamento 1 – Segunda parte*, donde se instalan tuberías, líneas eléctricas y de iluminación.

Luego, durante la operación de *Pintura,* se llevan a cabo diferentes tratamientos de protección y pintura, según el tipo de bloque procesado. Para esta operación se debe contar con estaciones de trabajo específicas y acondicionadas con el equipamiento necesario para dichas tareas. Al finalizar esta etapa, los bloques se encuentran en

condición de atravesar el último proceso de equipamiento, llamado *Prearmamento 2.* Aquí se colocan e instalan todos aquellos equipos que se pueden deteriorar durante el proceso de pintura, tales como instalaciones eléctricas y componentes electrónicos.

Finalmente, los bloques prefabricados, pintados y equipados se transportan y se colocan en el dique seco, para llevar a cabo la etapa de *Montaje final* a través operaciones de soldadura. Dichas operaciones se efectúan cumpliendo una secuencia específica de ensamblado, que se define según las especificaciones de diseño de la embarcación que se construye y teniendo en cuenta los diferentes tipos de unidades ensambladas: bloque base, bloque lateral o bloque superior (ver Figura 2.3). La construcción de una embarcación siempre comienza con un bloque base previamente establecido, luego, el resto de los bloques se ensamblan y sueldan respetando la secuencia mencionada. Como se puede observar en la Figura 2.3, el tiempo de procesamiento en la etapa de *Montaje en grada* también queda definido en función de los tipos de bloques a ensamblar (lateral o superior). Se debe tener en cuenta que un bloque puede ser ensamblado si las operaciones de montaje de su bloque predecesor han terminado. Por lo tanto, si un bloque es trasladado a la zona de grada antes de que finalicen las operaciones previas, tendrá un tiempo de espera antes de ser procesado.

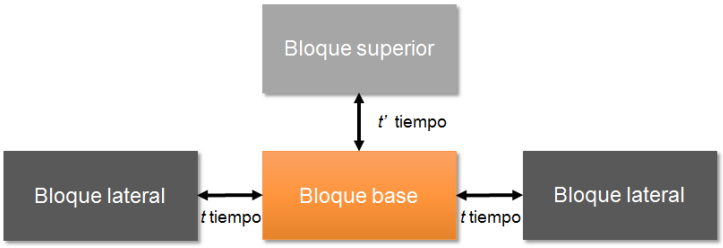



Figura 2.3. Esquema de montaje de bloques en grada.

## 2.2.2. Hipótesis y consideraciones generales

Como se describió anteriormente, el proceso de construcción naval basado en un diseño modular integrado implica el procesamiento de un conjunto de órdenes de trabajo

$i \in I$ a través de diferentes etapas $s \in S$, siguiendo una secuencia de producción predefinida y conocida. Los trabajos son asignados y secuenciados de manera consecutiva en dichas etapas, siguiendo la misma receta de procesamiento. Por lo tanto, el problema abordado en este capítulo trata sobre la programación detallada de las operaciones de un sistema productivo tipo flow shop flexible (FFSP) con múltiples etapas de procesamiento, cada una de las cuales cuenta con unidades de trabajo $k \in K_s$ en paralelo e idénticas entre sí, es decir, que no existen prioridades entre las mismas.

La estructura y el diseño del sistema son conocidos, así como también la composición modular de la embarcación que se va a construir (cantidad y tipo de bloques y sub-bloques). En la Figura 2.4 se ilustra un pequeño ejemplo de la configuración de un astillero como un entorno flow shop flexible con etapas de ensamblado, donde se procesan y se ensamblan $N = 8$ sub-bloques y $M = 4$ bloques para obtener una embarcación. Notar que, los sub-bloques que conforman un mismo bloque están representados con el mismo color. Se puede observar que cada bloque puede estar formado por uno o más sub-bloques. Por ejemplo, el bloque $i_{n+1}$ está formado por dos sub-bloques, $i_1$ e $i_2$, mientras que el bloque $i_{n+2}$ está formado sólo por el sub-bloque $i_3$.

El problema de programación planteado puede matemáticamente ser definido a partir de los siguientes conjuntos, donde se identifican sus elementos y respectivos índices:

- $S$ (índice $s = 1,2, \dots |S|$): Conjunto de todas las etapas de procesamiento, tanto de producción como de ensamblado, que componen el sistema de manufactura (astillero).

- $K$ (índice $k = 1,2, \dots |K|$): Conjunto de todas las celdas, talleres o estaciones de trabajo que forman parte del astillero, y son consideradas unidades de procesamiento.

- $I$ (índice $i = 1,2, \ldots |I|$): Conjunto de órdenes de productos que se elaboran siguiendo la misma secuencia de procesamiento.

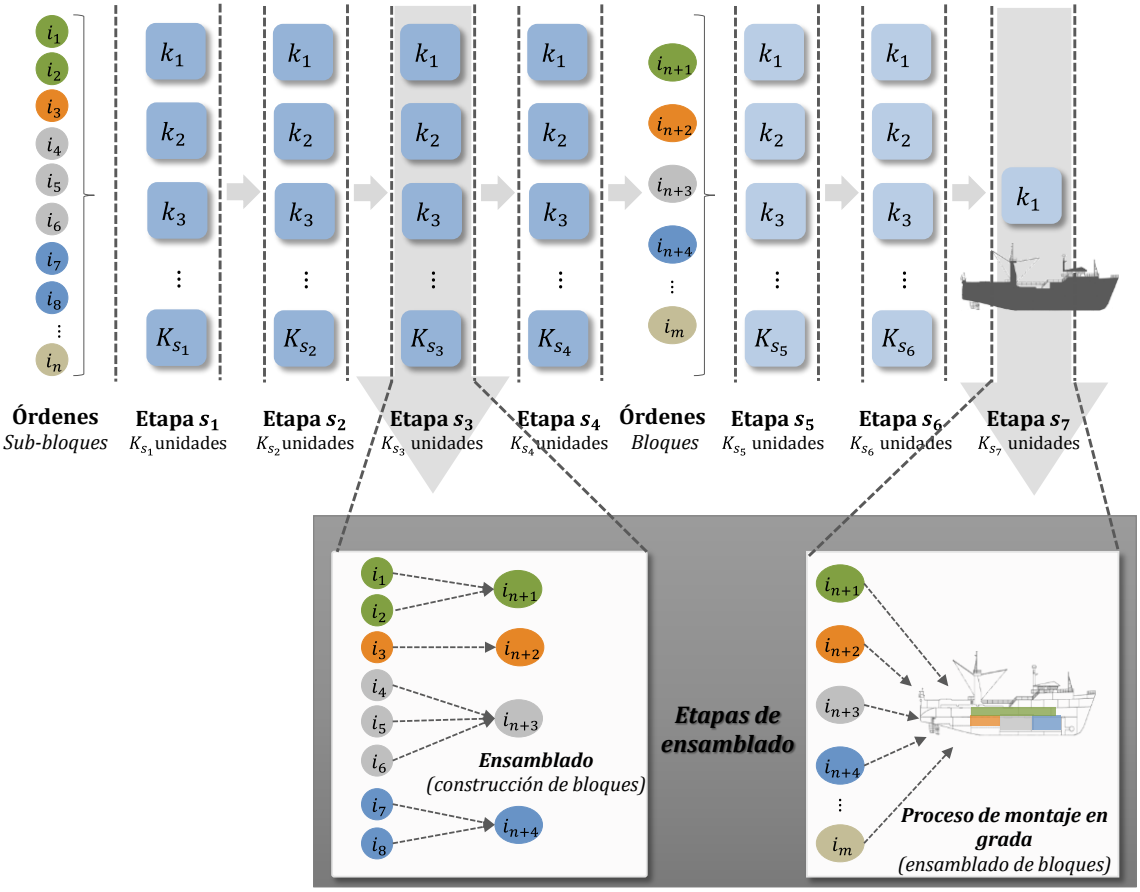



Figura 2.4. Proceso de construcción naval - FFSP con operaciones de ensamblado.

Como se mencionó anteriormente los bloques se construyen a partir de la elaboración de sub-bloques, por lo que el conjunto $I$ está compuesto por dos subconjuntos, $I^b$ y $I^{sb}$, tal que $I = I^b \cup I^{sb}$.

- $I^b$: Subconjunto de todos los bloques que componen el producto final (barco o buque), $I^b \subset I$.

- $I^{sb}$: Subconjunto de todos los sub-bloques que son ensamblados para obtener los bloques, $I^{sb} \subset I$.

Análogamente, se definen cuatro subconjuntos de etapas para modelar de manera diferenciada aquellas dedicadas al procesamiento de sub-bloques ($S^{sb}$) y bloques ($S^b$), de las que sólo realizan operaciones de ensamblado de los mismos ($S^a, S^a_{seq}$), donde se cumple que $S = S^{sb} \cup S^b$.

- $S^{sb}$: Subconjunto de etapas destinadas al procesamiento de sub-bloques, $S^{sb} \subset S$.
- $S^b$: Subconjunto de etapas que sólo procesan bloques, $S^b \subset S$.
- $S^a$ : Subconjunto de etapas intermedias dedicadas a operaciones de ensamblado de sub-bloques, $S^a \subset S^b$.
- $S^a_{seq}$: Subconjunto de etapas dedicadas al montaje del producto final, es decir, al ensamblado de bloques, $S^a_{seq} \subset S^b$.

Es importante aclarar que, la diferencia entre ambas etapas de ensamblado radica en que las primeras, $s \in S^a$, comienzan el ensamblado de un determinado producto una vez que se encuentran disponibles todas sus partes correspondientes (subensambles o sub-bloques) que lo conforman, en alguna unidad de dicha etapa; mientras que en las etapas $s \in S^a_{seq}$ se debe respetar una secuencia de ensamblado definida a través del parámetro $Seq_i$, es decir, se puede comenzar las operaciones de montaje de un dado producto a medida que se encuentren disponibles sus subensambles, siempre y cuando se respete su secuencia de montaje $Seq_i$. Notar que, para el problema detallado anteriormente, el conjunto $S^a$ está formado por la etapa $s$ encargada del ensamble de sub-bloques para obtener bloques, y $s \in S^a_{seq}$ corresponde a la última etapa del proceso productivo, cumpliéndose $S^a_{seq} = |S|$, donde se realiza el montaje de los bloques.

Además de las características mencionadas, a continuación se detallan las suposiciones planteadas para la formulación de los modelos matemáticos.

- Cada orden de producto $i \in I$ debe ser procesada siguiendo una secuencia predefinida de etapas de procesamiento y ensamblaje $s \in S$, con unidades idénticas $k \in K$ operando en paralelo.

- Cada orden de producto $i$ representa a un bloque o sub-bloque, por lo que sólo puede pertenecer a uno de los dos subconjuntos: $I^{sb} \subset I$ y $I^{b} \subset I$.

- Cada bloque $i \in I^b$ está compuesto por uno o más sub-bloques específicos $i' \in SB_i$.

- Al existir operaciones de ensamblado, los productos $i \in I$ no son procesados en todas las etapas $s \in S$.

- Cada unidad de procesamiento $k \in K$ tiene capacidad de procesar un sólo producto $i \in I$ por vez, y del mismo modo, cada producto $i$ no puede ser procesado de manera simultánea en dos o más unidades.

- Cada etapa $s \in S$ tiene $k \in K_s$ unidades idénticas en paralelo.

- Las primeras etapas $s \in S^{sb}$ sólo procesan sub-bloques, mientras que las últimas etapas $s \in S^{b}$ sólo procesan bloques.

- Los sub-bloques $i' \in SB_i$ son ensamblados en una etapa intermedia específica $s \in S^a$ para formar el bloque $i \in I^b$.

- Los bloques $i \in I^b$ son ensamblados en la etapa final del proceso de construcción $s \in S^a_{seq}$, donde $S^a_{seq} = |S|$, para obtener el producto final.

- La secuencia de ensamblado final que se debe cumplir en la última etapa del proceso se conoce a priori y está representada por el parámetro $Seq_i$.

- Todos los parámetros del modelo, como los tiempos de procesamiento $tp_{is}$, son conocidos a priori con total certidumbre y permanecen sin variaciones a lo largo del horizonte de programación.

- Los tiempos de transferencia de los productos entre las unidades de etapas consecutivas se suponen muy pequeños, comparados a los tiempos de procesamiento, y por lo tanto, son despreciables o bien, incluidos en éstos últimos.

- No hay restricciones en cuanto al abastecimiento de materia prima y materiales, en consecuencia, se considera una disponibilidad ilimitada de los mismos y el abastecimiento a las unidades de procesamiento se supone continuo.

- Pueden adoptarse dos políticas de almacenamiento intermedio: (i) sin almacenamiento intermedio (NIS) y (ii) almacenamiento intermedio ilimitado (UIS). En el primer caso, se considera que no existen lugares específicos en los talleres del astillero para el almacenamiento de bloques y sub-bloques, sin embargo, se admite que pueden permanecer en la unidad donde fueron procesados hasta poder ser transferidos a la unidad asignada para la siguiente operación. Mientras que, en el segundo caso, se considera que existe suficiente espacio para el almacenamiento de los dichos productos, de tal manera que no sea una restricción a tener en cuenta.

De acuerdo a las hipótesis enunciadas y sobre la base de los siguientes datos del problema:

- la configuración del sistema productivo, haciendo referencia a la cantidad de etapas productivas de sub-bloques $S^{sb}$ y bloques $S^{b}$, las etapas dedicadas a operaciones de ensamblado ($S^{a}$, $S^{a}_{seq}$) y el número de unidades de procesamiento $k \in K_s$ de cada una de ellas;

- el diseño modular del producto final (barco), es decir, la cantidad y el tipo de bloques $i \in I^b$, y cantidad de sub-bloques que componen a cada uno de ellos, $i' \in SB_i$;
- los tiempos de procesamiento de los productos $i$ en cada etapa $s$ ($tp_{is}$);
- la política de almacenamiento intermedio adoptada;
- la secuencia final de ensamblado de bloques en grada, correspondiente al montaje del producto final;

se desea determinar el programa detallado de la producción a través de la aplicación de diferentes enfoques exactos de optimización, para brindar la solución óptima con respecto a:

- ✓ La asignación de las tareas de producción a los recursos disponibles, es decir, la asignación de las órdenes de producto a las unidades de cada etapa del proceso.
- ✓ La secuenciación de las órdenes a procesar en cada taller del astillero.
- ✓ Tiempos de inicio y finalización del procesamiento de todos los productos a elaborar y ensamblar en el astillero.
- ✓ Tiempo total requerido para la fabricación del producto final, es decir, para la construcción de la embarcación bajo estudio.

## 2.3. Formulaciones matemáticas

Habiendo definido el proceso de construcción y montaje de una embarcación de tamaño industrial como un problema de programación de operaciones de tipo flow shop flexible (FFSP), y con el objetivo de representar matemáticamente las restricciones de dicho problema, se presentan tres formulaciones alternativas, que consisten en modelos de programación matemática lineal mixta-entera (MILP) basados en los siguientes

conceptos: (i) intervalos de tiempo o "time-slots", (ii) precedencia inmediata y (iii) precedencia general global. Dichas formulaciones han sido reportadas inicialmente en la literatura, por Pinto y Grossmann (1995), Cerdá et al. (1997) y Méndez et al. (2001), respectivamente, para la programación detallada a corto plazo de plantas tipo "batch" de múltiples etapas.

Asimismo, dichas formulaciones se sustentan en cuatro conceptos importantes: (i) el uso de una representación continua del tiempo, (ii) el desacoplamiento de las decisiones de asignación y secuenciación, (iii) la disponibilidad ilimitada de recursos, a excepción de las unidades de procesamiento, y (iv) el modelado de operaciones de ensamblado en etapas específicas.

De acuerdo a lo expuesto, a partir de las propuestas originales de la literatura, se introducen los cambios necesarios para incorporar las restricciones operacionales adicionales y asegurar el comportamiento real de los procesos de ensamblado de los diferentes componentes (o productos intermedios) involucrados en la construcción de barcos.

El objetivo principal de los modelos MILP propuestos es determinar la mejor solución al problema FFSP brindando el programa detallado de todas las operaciones del proceso productivo bajo estudio, de manera tal que se minimice el tiempo total requerido para el procesamiento y ensamblaje de todos los productos involucrados en la fabricación de una embarcación. Esto se debe a que las compañías navales buscan aumentar su eficiencia productiva, y con esto su ganancia, la cual está directamente relacionada al tiempo que se necesita para construir una embarcación a gran escala. Por lo tanto, para el objetivo de optimización se utiliza el criterio del makespan, representado por la restricción (2.1). Se define una variable continua $MK$ que determina el tiempo total necesario para realizar todas las operaciones relacionadas a la obtención del producto final.

$$minimizar\ MK \tag{2.1}$$

### 2.3.1. Formulación MILP basada en "time-slots"

El primer modelo MILP de tiempo continuo que se plantea para la resolución del problema específico de programación de la producción en el proceso de ensamblado en bloques está basado en "time-slots". Bajo este concepto, los slots representan intervalos de tiempo de longitud variable, donde se asignan las órdenes de producto, que en este caso son bloques o sub-bloques. El número de slots postulados en cada unidad de cada etapa de procesamiento puede diferir uno de otro, por lo que su representación resulta asincrónica. Además, se debe tener en cuenta que a lo sumo se procesa un producto en cada slot de una determinada unidad, sin embargo, su longitud se vuelve nula si no se le asigna ninguna actividad de procesamiento.

En general, en las representaciones basadas en "time-slots", la cantidad de slots propuestos para cada unidad es una decisión clave debido a que no se conoce a priori y puede incidir en el espacio de búsqueda, la optimalidad del modelo y su rendimiento computacional (Fumero et al., 2012a; Ierapetritou y Floudas, 1998; Lim y Karimi, 2003; Pinto y Grossmann, 1995b). Definir pocos "time-slots" suele comprometer la optimalidad de la solución, y por otra parte, una gran cantidad de los mismos afecta directamente la performance computacional del modelo. Por lo tanto, para la formulación basada en el concepto de "time-slots" presentada en este capítulo, se propone una expresión particular de manera de incrementar su rendimiento computacional, sin perder de vista la optimalidad. Al mismo tiempo, se considera la existencia de operaciones de ensamblado y, por consiguiente, que no todas las etapas deben procesar la misma cantidad de órdenes de productos, lo cual afecta directamente al número de slots postulados.

Por otro lado, se definen variables binarias y continuas que se detallan a continuación, junto con cada una de las restricciones que integran el modelo matemático:

- $W_{ipk}$: variable binaria que relaciona la orden de producto con los intervalos de tiempo y las unidades de procesamiento, tomando valor 1 cuando el producto $i \in I$ (bloque o sub-bloque) es asignado al slot $p \in P_k$ de la unidad de procesamiento $k \in K$.

- $Ts_{is}$: variable continua positiva que computa el tiempo en que se inicia el procesamiento del producto $i$ en cada etapa $s$.

- $Tf_{is}$: variable continua positiva que define el tiempo de finalización de las actividades de procesamiento y ensamblado del producto $i$ en la etapa $s$.

- $Ts_{pk}$: variable continua positiva utilizada para reflejar el tiempo de inicio del procesamiento del slot $p$ en cada etapa $s$.

- $Tf_{pk}$: variable continua positiva que define el tiempo de finalización del procesamiento del slot $p$ en la etapa $s$.

**Restricciones de asignación.** Cada orden de producto debe ser asignada en cada etapa, a un slot de una unidad de procesamiento que pertenezca a dicha etapa. La restricción (2.2) expresa la asignación mencionada a través del uso de la variable $W_{ipk}$, asegurando que cada bloque $i \in I^b$ se procesa en cada etapa $s \in S^b$ en exactamente un "time-slots" $p$ de alguna unidad $k \in K_s$. Del mismo modo, se cumple para cada sub-bloque $i \in I^{sb}$ procesado en cada etapa $s \in S^{sb}$.

$$\sum_{k \in K_s} \sum_{p \in P_k} W_{ipk} = 1 \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^b, s \in S^b \right) \right) \tag{2.2}$$

Por otro lado, teniendo en cuenta que la variable $W_{ipk}$ define la relación producto-slot, la misma se utiliza nuevamente para definir la restricción (2.3) y establecer que cada "time-slots" $p \in P_k$ de la unidad $k \in K_s$, correspondiente a la etapa $s$, se puede asignar como máximo a una orden de producto $i \in I$.

$$\sum_{\substack{(i\in I^{sb},s\in S^{sb})\\ \cup(i\in I^{b},s\in S^{b})}} W_{ipk} \leq 1 \qquad \forall s \in S, k \in K_s, p \in P_k \tag{2.3}$$

Asimismo, no debe haber posiciones vacías entre los intervalos de tiempo $p$ y $p+1$ que pertenecen a una misma unidad de procesamiento $k$. En consecuencia, la restricción (2.4) asegura que, para cada unidad $k$, el slot $p+1$ sólo puede ser utilizado si el slot $p$ ha sido ya asignado para procesar una orden de producto.

$$\sum_{i\in I} W_{i(p+1)k} \leq \sum_{i\in I} W_{ipk} \qquad \forall k \in K, p \in P_k, (p+1) \in P_k \tag{2.4}$$

**Restricciones de tiempo.** Para poder modelar las decisiones de tiempo se utilizan las cuatro variables continuas detalladas anteriormente, que representan los tiempos de inicio y finalización del procesamiento de las órdenes de producto ($Tf_{is}, Ts_{is}$) y de los "time-slots" ($Tf_{pk}$ y $Ts_{pk}$). Los valores de dichas variables son determinados por las restricciones (2.5)-(2.7). El tiempo de finalización de la orden $i$ en la etapa $s$, $Tf_{is}$, y el tiempo de finalización del "time-slots" $p$ en la unidad $k$, $Tf_{pk}$, se calculan mediante las restricciones (2.5) y (2.6), respectivamente. Esto se debe a que el tiempo de inicio más el tiempo de procesamiento $tp_{is}$ están asociado a la orden asignada al intervalo de tiempo $p$.

$$Tf_{is} \geq Ts_{is} + tp_{is} \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right) \tag{2.5}$$

$$Tf_{pk} \geq Ts_{pk} + \sum_{\substack{(i\in I^{sb},s\in S^{sb})\\ \cup(i\in I^{b},s\in S^{b})}} W_{ipk} tp_{is} \qquad \forall s \in S, k \in K_s, p \in P_k \tag{2.6}$$

Una orden no puede comenzar a procesarse en la etapa $s$ hasta que su procesamiento haya finalizado en la etapa anterior $(s-1)$, condición expresada a través de las restricciones (2.7a) o (2.7b), según la política de almacenamiento adoptada en el sistema

productivo. La primera de ellas contempla una política de almacenamiento intermedio tipo NIS, mientras que la segunda tiene en cuenta un almacenamiento sin restricciones, tipo UIS.

$$Ts_{is} = Tf_{i(s-1)} \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right) : s > 1 \qquad (2.7a)$$

$$Ts_{is} \geq Tf_{i(s-1)} \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right) : s > 1 \qquad (2.7b)$$

**Restricciones de ensamblado.** Debido a que en las etapas de ensamblado ingresa una determinada cantidad de órdenes, asociadas a productos con un determinado valor agregado (como los sub-bloques o bloques, según corresponda), para ser procesadas y obtener, de este modo, una cantidad menor de productos, pero de mayor valor agregado, dichas etapas deben ser modeladas de manera diferente. En el caso de la primera etapa de ensamblado, un bloque $i$ sólo puede comenzar a construirse cuando los sub-bloques $i' \in SB_i$ que lo conforman han terminado su procesamiento en la etapa previa. Notar que, siempre se cumple que $\left|I^b\right| \leq \left|I^{sb}\right|$. Esto se modela a través de la siguiente expresión:

$$Ts_{is} \geq Tf_{i'(s-1)} \qquad \forall\, i \in I^b, i' \in SB_i, s \in S^a \qquad (2.8)$$

Por otra parte, en la etapa de *Montaje final*, $s = |S|$ siendo que $s \in S^a_{seq}$, se llevan a cabo las operaciones de ensamblado de los bloques según la secuencia final de montaje preestablecida. Dicha secuencia, definida por el parámetro $Seq_i$, coincide con el orden en que se definen los elementos del conjunto $I^b$, que corresponden a las órdenes de productos asociadas a los bloques. Esta operación se satisface a través de las siguientes restricciones:

$$TF_{i's} \geq TF_{is} \qquad \forall\, (i, i') \in I^b, s \in S^a_{seq} : Seq_i < Seq_{i'} \qquad (2.9)$$

$$W_{ipk} - W_{i',p+1,k} = 0 \qquad (2.10)$$

$$\forall\,(i,i') \in I^b, k \in K_s, s \in S^a_{seq}, p \in P_k, (p+1) \in P_k : Seq_i < Seq_{i'}$$

Notar que, la restricción de ensamblado final se puede modelar a través del uso de variables binarias o continuas, utilizando la restricción (2.9) o (2.10), respectivamente.

Siguiendo el ejemplo ilustrativo de la Figura 2.4, en la Figura 2.5 se representan gráficamente las restricciones (2.8) y (2.9). Un aspecto importante que refleja este diagrama es la importancia del cumplimiento de las restricciones (2.8)-(2.10) para modelar las tareas de ensamblado presentes en el sistema bajo estudio: una etapa intermedia ($s_3$), donde se deben ensamblar sub-bloques, y la última etapa ($s_7$), donde el orden en que se debe efectuar el montaje del barco debe ser respetado estrictamente.

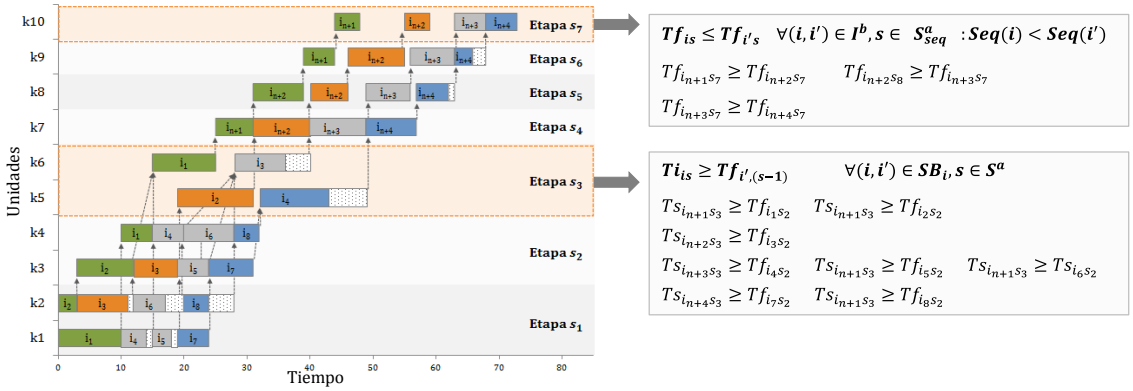



Figura 2.5. Diagrama Gantt del procesamiento de 8 sub-bloques y 4 bloques.

**Restricciones de secuenciación.** Se define la restricción de secuencia de slots a través de la restricción (2.11), que determina que un slot $(p+1)$ no puede comenzar su procesamiento en la unidad $k$ hasta que el slot previo $p$ haya terminado su procesamiento en la misma unidad.

$$Tf_{pk} \leq Ts_{(p+1)k} \qquad \forall\, k \in K, p \in P_k, (p+1) \in P_k \qquad (2.11)$$

Se debe tener en cuenta que si la orden $i$ es asignada al slot $p \in P_k$ de la unidad de procesamiento $k \in K_s$, debe cumplirse que $Tf_{is} = Tf_{pk}$ y $Ts_{is} = Ts_{pk}$. Estas condiciones son modeladas por medio de las restricciones (2.12)-(2.15), conocidas como restricciones Big-M.

$$Tf_{is} - Tf_{pk} \geq -M(1 - W_{ipk}) \qquad (2.12)$$
$$\forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right), k \in K_s, p \in P_k$$

$$Tf_{is} - Tf_{pk} \leq M(1 - W_{ipk}) \qquad (2.13)$$
$$\forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right), k \in K_s, p \in P_k$$

$$Ts_{is} - Ts_{pk} \geq -M(1 - W_{ipk}) \qquad (2.14)$$
$$\forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right), k \in K_s, p \in P_k$$

$$Ts_{is} - Ts_{pk} \leq M(1 - W_{ipk}) \qquad (2.15)$$
$$\forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right), k \in K_s, p \in P_k$$

**Cálculo del número de slots propuestos.** Como ya se ha mencionado, el número de "time-slots" $P_k$ definidos para cada unidad de procesamiento $k$ afecta directamente la performance computacional del modelo matemático. Asimismo, es un dato que no se conoce de antemano, por lo tanto, debe ser seleccionado adecuadamente de manera de asegurar la optimalidad de la solución reportada por el modelo matemático.

En la literatura, se han propuesto diferentes expresiones que varían en las decisiones de asignación de cada modelo. Particularmente, en el problema abordado en este capítulo, existen etapas de ensamblado de productos, lo que se traduce en que las primeras etapas procesan un mayor número de productos y por consiguiente, tendrán un mayor número de slots postulados que las últimas, indistintamente de la expresión que se proponga para su cálculo. Teniendo en cuenta esto, y que las unidades en paralelo de cada etapa son idénticas, se podría suponer el número máximo de slots necesarios para procesar todas las órdenes en cada unidad $k$: $P_k = \left|I^{sb}\right|$ para $k \in K_s, s \in S^{sb}$ y $P_k = \left|I^{b}\right|$ para $k \in K_s, s \in S^{b}$. Sin embargo, esta propuesta podría ser mejorada reduciendo la cantidad de slots de manera que el modelo sea computacionalmente más eficiente.

Notar que, el mínimo número de slots que se puede postular, considerando que los tiempos de procesamiento de todos los productos son similares para un dada etapa de procesamiento y que se distribuyen de manera equitativa entre las unidades en paralelo de dicha etapa, puede calcularse como: $P_k = \left|I^{sb}\right|/|K_s|$ y $P_k = \left|I^{b}\right|/|K_s|$, según se procesen bloques o sub-bloques en la etapa $s$. Pese a que este cálculo puede resultar muy provechoso desde el punto de vista del esfuerzo computacional requerido, resulta arriesgado debido que, al no poder asegurar similitud entre los tiempos de procesamiento, se podría perder la optimalidad de la solución hallada.

Otra propuesta que reduce la cantidad de slots postulados, pero asegura la optimalidad, tiene como hipótesis que al menos un slot es utilizado en todas las unidades de procesamiento $k \in K_s$ de cada etapa $s$. De este modo, el número máximo de slots considerados para cada unidad estaría dado por las restricciones (2.16a) y (2.16b), según corresponda. Estas expresiones son aplicadas en la presente formulación matemática para ajustar el número de slots propuesto para cada unidad.

$$P_k = \left|I^{sb}\right| - (|K_s| - 1) \qquad \forall\, k \in K_s, s \in S^{sb} \tag{2.16a}$$

$$P_k = \left|I^b\right| - (|K_s| - 1) \qquad \forall\, k \in K_s, s \in S^b \tag{2.16b}$$

Por otro lado, también se considera la expresión para el cálculo del número de slots propuesta por Fumero et al. (2012), la cual supone que las unidades $k \in K_s$, correspondientes a una misma etapa, se usan en orden ascendente. Dicha propuesta es adaptada al problema FFSP planteado a través de las restricciones (2.17a) y (2.17b). Además, tiene en cuenta la posición de cada unidad dentro de una etapa, lo que permite postular distintos números de slots a las unidades de una misma etapa, y con esto reducir aún más su cantidad. Cabe aclarar que, se aplican ambos conjuntos de restricciones a fin de poder comparar el desempeño computacional que presenta el modelo frente a las dos propuestas.

$$P_k = \frac{|I^{sb}| - (|K_s| - k)}{k} \qquad \forall k \in K_s, s \in S^{sb} \tag{2.17a}$$

$$P_k = \frac{|I^{b}| - (K_s - k)}{k} \qquad \forall k \in K_s, s \in S^{b} \tag{2.17b}$$

**Cálculo del makespan.** Como se mencionó anteriormente y se mostró a través de la restricción (2.1), el objetivo del problema es minimizar el valor del makespan $MK$, el cual se determina a través de la siguiente desigualdad:

$$MK \geq Tf_{is} \qquad \forall i \in I, s \in S \tag{2.18}$$

## 2.3.2. Formulación MILP basada en precedencia inmediata

Con el objetivo de representar matemáticamente las operaciones de producción y ensamblado de bloques y sub-bloques bajo el concepto de precedencia inmediata, se definen tres conjuntos diferentes de variables binarias:

- $YF_{ik}$: indica si la orden de producto $i$ (bloque o sub-bloque) es la primera en ser procesada en la unidad $k$.

- $Y_{ik}$: determina si la unidad de procesamiento $k$ procesa la orden $i$, pero no en primer lugar.

- $W_{ii'k}$: indica si la orden de producto $i'$ es procesada inmediatamente después de la orden $i$ en la unidad $k$.

A continuación, se agrupan y describen las restricciones utilizadas en esta formulación con representación de tiempo continuo, según se trate de: (i) asignación de órdenes a unidades de procesamiento, (ii) precedencia inmediata de órdenes, es decir, secuenciación de tareas en cada celda de trabajo o unidad (iii) cálculo de los tiempos de inicio y fin de las órdenes en cada etapa productiva, (iv) modelado de operaciones de ensamblado.

**Restricciones de asignación.** Cada sub-bloque $i \in I^{sb}$ se procesa en una sola unidad $k \in K_s$ en cada etapa $s \in S^{sb}$, y del mismo modo, cada bloque $i \in I^b$ se procesa en una única unidad $k \in K_s$ en cada etapa $s \in S^b$. Se formula la restricción (2.19) para establecer la asignación de cada orden de producto (bloque o sub-bloque) a una unidad de cada etapa correspondiente, donde puede ser procesada en primer lugar ($YF_{ik} = 1, Y_{ik} = 0$) o en otra posición de la secuencia ($YF_{ik} = 0, Y_{ik} = 1$).

$$\sum_{k \in K_s} (Y_{ik} + YF_{ik}) = 1 \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^b, s \in \left( S^b \cup S^a \right) \right) \right) \tag{2.19}$$

Además, como máximo puede haber una primera orden a ser procesada en cada unidad, condición que se impone a través de la siguiente restricción:

$$\sum_{i \in I} YF_{ik} \leq 1 \qquad \forall\, k \in \mathrm{K} \tag{2.20}$$

**Restricciones de precedencia directa.** Para las decisiones de precedencia de trabajos a ser procesados en una misma unidad $k$, se definen dos restricciones, (2.21) y (2.22), donde la variable binaria $W_{ii'k}$ sólo puede estar activa, es decir tomar valor 1, si el par de órdenes $(i, i')$ se procesan en la misma unidad $k$ (en ese caso, $Y_{ik} + YF_{ik} = 1$ y $Y_{i'k} = 1$).

$$W_{ii'k} \leq Y_{ik} + YF_{ik} \qquad \forall (i, i') \in I, k \in K: i \neq i' \tag{2.21}$$

$$W_{ii'k} \leq Y_{i'k} \qquad \forall (i, i') \in I, k \in K: i \neq i' \tag{2.22}$$

Por otra parte, como se ha mencionado, se debe modelar que una orden $i$ puede ser procesada en una unidad $k$, ya sea en primer lugar o después de otra orden $i'$, pero al mismo tiempo, se debe tener en cuenta que cuando una orden de producto se asigna a una unidad $k$ ($Y_{ik} = 1$), pero no es la última en ser procesada, debe tener un predecesor directo $i'$ en la secuencia de procesamiento ($W_{i'ik} = 1$, para algún $i' \in I$). De este modo, se presentan las restricciones (2.23) y (2.24) que establecen, respectivamente, las relaciones de precedencia detalladas.

$$YF_{ik} + \sum_{i' \in I: i \neq i'} W_{i'ik} \leq 1 \qquad \forall\, i \in I, k \in K \tag{2.23}$$

$$\sum_{i' \in I: i \neq i'} W_{i'ik} = Y_{ik} \qquad \forall\, i \in I, k \in K \tag{2.24}$$

Además, la restricción (2.25) afirma que, como máximo, una única orden de producto puede programarse inmediatamente después de otra orden en una determinada unidad, a menos que ésta sea la última en la secuencia de procesamiento de dicha unidad.

$$\sum_{i' \in I: i \neq i'} W_{ii'k} \leq 1 \qquad \forall\, i \in I, k \in K \tag{2.25}$$

**Restricciones de tiempo.** Como se mencionó anteriormente, las variables continuas no negativas $Tf_{is}$ y $Ts_{is}$ representan los tiempos iniciales y finales de procesamiento de cada orden de producto $i$ a ser procesada en la unidad $k \in K_s$ de la etapa $s$, respectivamente. La relación entre ambas variables queda definida por la restricción (2.5), donde se calcula el tiempo de finalización de $i$ en la etapa $s$ como la suma de su tiempo de inicio y el tiempo de procesamiento asociado a dicha etapa, $tp_{is}$.

En función de la política de almacenamiento entre etapas sucesivas establecida por el astillero (UIS o NIS), se emplea la restricción (2.7a) ó (2.7b), anteriormente definidas en la formulación MILP basada en "time-slots".

**Restricciones de ensamblado.** Para restringir el comienzo del armado de los bloques en la etapa de ensamblado $s \in S^a$, a la culminación del procesamiento de todos aquellos sub-bloques asociados a un mismo bloque en la etapa previa, y además, cumplir con la secuencia final de ensamblado de bloques en la etapa $s \in S^a_{seq}$, se utilizan las restricciones (2.8) y (2.9), respectivamente. Notar que, al igual que en la formulación MILP basada en "time-slots", el montaje de bloques que se lleva a cabo en la última etapa, también se puede modelar con las variables binarias de secuenciación a través de la restricción (2.26).

$$W_{i,i',k} = 1 \qquad \forall\, (i,i') \in I^b, k \in K_s, s \in S^a_{seq} : i < \left|I^b\right|, Seq_{i'} = Seq_i + 1 \tag{2.26}$$

**Restricciones de secuenciación**. Por último, las restricciones de secuenciación que se deben cumplir entre las diferentes órdenes de productos procesadas en cada unidad $k$, quedan representadas por la restricción (2.27), donde se establece que el tiempo de inicio $Ts_{i's}$ de la orden $i'$ en la etapa $s$ debe ser mayor que el tiempo final $Tf_{i,s}$ de su predecesor inmediato $i$, siempre que se cumpla que ambos se procesan en la misma $k \in K_s: W_{ii'k} = 1$.

$$Ts_{i's} \geq Tf_{i,s} - M(1 - W_{ii'k}) \tag{2.27}$$

$$\left(\left((i,i') \in I^{sb}, s \in S^{sb}\right) \cup \left((i,i') \in I^{b}, s \in S^{b}\right)\right), k \in K_s: i \neq i'$$

**Cálculo del makespan.** La función objetivo para esta formulación, al igual que para la formulación MILP antes presentada, es minimizar el makespan a través de la restricción (2.18) ya definida.

### 2.3.3. Formulación MILP basada precedencia general global

En esta subsección se propone una formulación matemática MILP, también con manejo continuo del tiempo, pero basada en el enfoque de precedencia general global para la resolución del problema de ensamblado de bloques descripto. La noción generalizada de precedencia está basada en el concepto de que una tarea, operación o trabajo precede a otro siempre que sea ejecutado más temprano en el mismo equipo o unidad de procesamiento, relajando el enfoque de precedencia inmediata que obliga a que la variable binaria de secuenciación tome valor 1 ($W_{ii'k} = 1$), si y sólo si, el trabajo $i$ es predecesor inmediato del trabajo $i'$.

Al igual que en el caso anterior, se utilizan variables binarias para las decisiones de asignación y secuenciación. Sin embargo, el concepto de precedencia general reduce el número de variables binarias referidas a la secuenciación de las tareas de procesamiento. Esta reducción se obtiene definiendo dicha variable binaria sólo para los pares de

productos $(i, i')$, donde $i < i'$. De este modo, dicho concepto reduce significativamente la cantidad de variables de secuenciación cuando se lo compa, por ejemplo, con el enfoque de precedencia inmediata. Además, la variable $W_{ii'k}$ toma significado sólo cuando ambas órdenes o trabajos $(i, i')$ son asignados a la misma unidad de procesamiento, es decir, cuando se activan las variables $Y_{ik} = Y_{i'k} = 1$. En este caso, $W_{ii'k}$ toma valor 1 siempre que la orden de producto $i$ se procese antes de la orden de producto $i'$ en la secuencia de procesamiento de la unidad $k$, en caso contrario toma valor 0, indicando que $i'$ se procesa antes que $i$ en la unidad $k$. Si ambas ordenes no son asignadas a la misma unidad, el valor que de $W_{ii'k}$ es despreciable. De este modo, se definen las siguientes dos variables binarias:

- $W_{ii'k}$: determina la secuencia de procesamiento (producción o ensamblado) de trabajos en cada unidad $k$.
- $Y_{ik}$: define la asignación de órdenes de productos (bloques y sub-bloques) a las unidades $k$.

A continuación, se describen los conjuntos de restricciones que integran la presente formulación matemática.

**Restricciones de asignación.** La restricción (2.28) define la restricción de asignación de sub-bloques y bloques a las unidades de cada etapa del proceso de construcción naval, donde una orden de producto (sub-bloque y bloque) debe procesarse sólo en una unidad de cada etapa.

$$\sum_{k \in K_s} Y_{ik} = 1 \qquad \forall \left( \left( i \in I^{sb}, s \in S^{sb} \right) \cup \left( i \in I^{b}, s \in S^{b} \right) \right) \tag{2.28}$$

**Restricciones de tiempo**. Al igual que en la formulación basada en el concepto de "time-slots", la restricción (2.5) se utiliza para calcular los tiempos de inicio y fin de procesamiento de cada orden en cada etapa del proceso productivo. También se emplean

las restricciones (2.7a) o (2.7b), según corresponda, para modelar la política de almacenamiento.

**Restricciones de ensamblado.** Se incluyen las restricciones (2.8) y (2.9), utilizadas en las dos formulaciones anteriores, para modelar las operaciones de ensamblado, tanto de sub-bloques como de bloques. Asimismo, la secuencia final de ensamblado de bloques se puede modelar utilizando la variable binaria de precedencia a través de la restricción (2.29).

$$W_{ii'k} = 1 \qquad \forall\, (i,i') \in I^{sb}, k \in K_s, s \in S^a_{seq}: Seq_i < Seq_{i'} \tag{2.29}$$

**Restricciones de secuenciación**. La secuenciación de órdenes de productos en la misma unidad $k$ están dadas por las restricciones (2.30) y (2.31). La restricción (2.30) determina el tiempo a partir del cual se puede iniciar el procesamiento de una tarea $i'$ en la etapa de procesamiento $s$ ($Ts_{i's}$), el cual debe ser mayor al tiempo final de procesamiento $Tf_{is}$ de todos los predecesores que tenga en dicha etapa, es decir, $W_{ii'k} = 1$ para algún $k \in K_s$ tal que $Y_{ik} = Y_{i'k}$=1, con $i < i'$. Si $W_{ii'k} = 0$, la restricción (2.30) se transforma en redundante y se activa (2.31).

$$Ts_{i's} \geq Tf_{is} - M(1 - W_{ii'k}) - M(2 - Y_{ik} - Y_{i'k}) \tag{2.30}$$

$$\forall \Big( \big(i \in SB, i' \in SB, s \in S^{sb}\big) \cup \big(i \in B, i' \in B, s \in S^{b}\big) \Big), k \in K_s, i < i'$$

$$Ts_{is} \geq Tf_{i's} - MW_{ii'k} - M(2 - Y_{ik} - Y_{i'k}) \tag{2.31}$$

$$\forall \Big( \big(i \in SB, i' \in SB, s \in S^{sb}\big) \cup \big(i \in B, i' \in B, s \in S^{b}\big) \Big), k \in K_s, i < i'$$

**Cálculo del makespan.** La función objetivo del modelo MILP se define por medio de las restricciones (2.1) y (2.18) definidas anteriormente en este capítulo.

## 2.4. Aplicación de los modelos matemáticos a diferentes instancias de un caso de estudio

En esta sección, se utilizan las formulaciones presentadas para dar solución a un problema complejo de optimización de una empresa europea referente a nivel mundial en el diseño y construcción de buques militares y buques civiles de alta tecnología. La compañía, dedicada a exportar sus productos y diseños a algunos de los países más avanzados del mundo, cuenta con equipos especializados que despliegan la fabricación de diversos tipos de buques. En particular, se considera como caso de estudio para el presente capítulo, la construcción de un buque militar de grandes dimensiones.

Se introducen varias instancias del caso de estudio a fin de evaluar el desempeño de cada formulación matemática. De este modo, la complejidad del problema abordado en relación a la estructura del astillero, donde se lleva a cabo el proceso de manufactura, y al diseño modular de la embarcación que se desea construir, es gradualmente incrementada de manera de poder comparar la performance computacional de los diferentes métodos rigurosos expuestos.

A continuación, se describen las características que tiene el caso de estudio y las diferentes instancias evaluadas a partir de los enfoques de optimización. Luego, se muestra la complejidad intrínseca de los problemas abordados y las estadísticas reportadas por dichos enfoques.

Los modelos MILP son implementados en el lenguaje de modelado GAMS 24.9.2. En todos los casos, se emplea el optimizador de modelos matemáticos lineales mixto-entero CPLEX 12.7.1.0 para resolver los problemas y además, reportar el tiempo computacional requerido para hallar la solución óptima en cada caso. Los modelos se corren una PC con procesador Intel Xeon X5650 de cuatro núcleos (2,6 GHz) y 32 GB de RAM. El criterio de terminación utilizado en todas las instancias es de una diferencia de optimalidad o GAP del 0%, asegurando la optimalidad de los modelos, ó 3600 segundos de tiempo de CPU.

### 2.4.1. Definición del caso de estudio

El caso de estudio considerado cuenta con 7 etapas de procesamiento, de las cuales 2 están destinadas a operaciones de ensamblado: la primera ensambla sub-bloques para formar los grandes bloques y la segunda, ensambla bloques para obtener un barco de tamaño industrial. Tal como se muestra en la Figura 2.6, cada etapa tiene una cantidad $K_s$ de unidades idénticas operando en paralelo, sumando un total de 42 unidades de procesamiento. Como se puede apreciar, la última etapa ($s_7$) está formada por un sola unidad que representa la grada o dique seco, donde se llevan a cabo las operaciones finales de montaje de bloques.

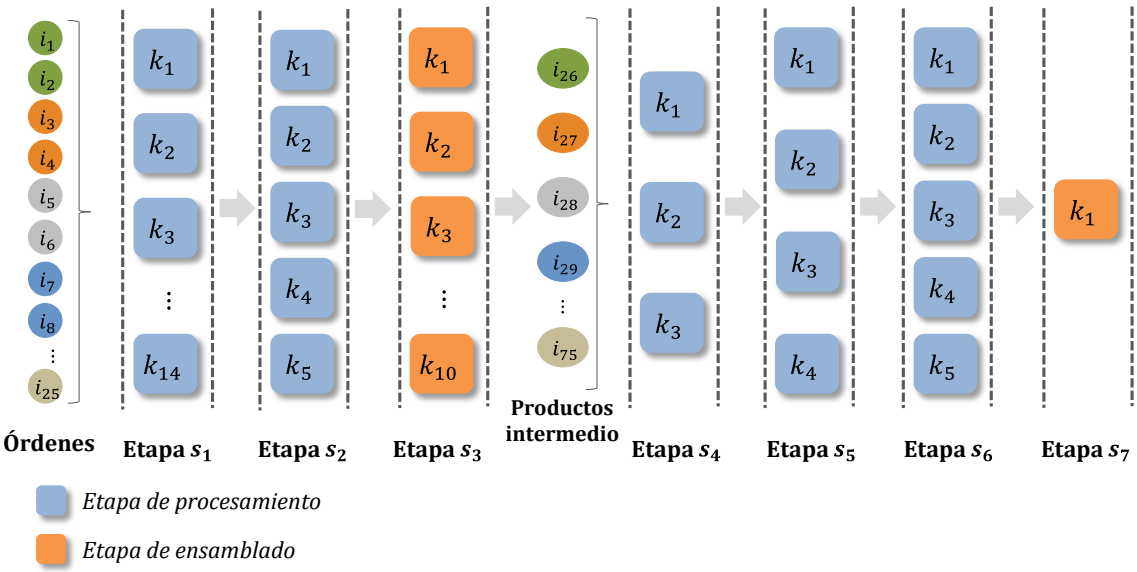



Figura 2.6. Estructura del proceso de construcción de barcos.

La embarcación a fabricar se compone de 25 bloques, cada uno de los cuales se encuentra conformado por dos sub-bloques con características similares. Además, se debe tener en cuenta que los tiempos de procesamiento varían en cada etapa productiva, de acuerdo al tipo de bloque a procesar. Como se puede observar en la Figura 2.7, se pueden distinguir 5 tipos diferentes de bloques, según a qué parte del barco pertenezcan: cabinas, proa, popa, fondos, cubiertas y compartimentos, respectivamente. De este modo, el conjunto $I$ está integrado por 75 elementos (ver Figura 2.6), 25 correspondientes al

conjunto de bloques y 50 al conjunto de sub-bloques ($I = I^b \cup I^{sb}$). Cabe aclarar que, por razones de confidencialidad, no se publican en esta tesis los tiempos de procesamiento de las órdenes de producto $i \in I$ en las diferentes etapas de manufactura que conforman al astillero.

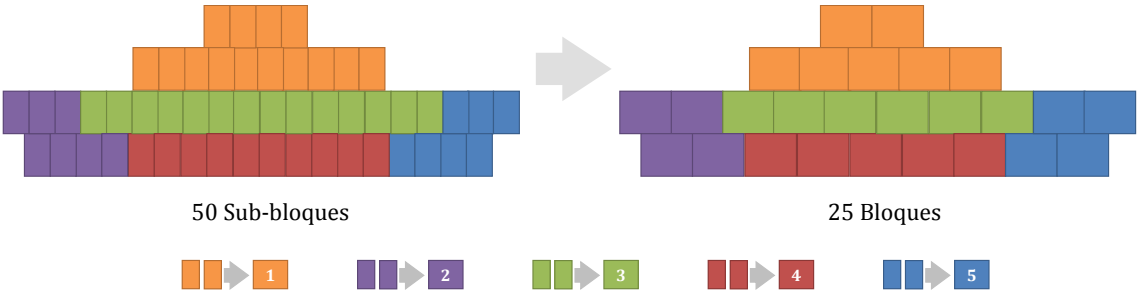



Figura 2.7. Estrategia constructiva de un barco formado por 25 bloques y bloques constituidos por 2 sub-bloques.

A los efectos de apreciar el desempeño computacional de las formulaciones propuestas para obtener el "schedule" óptimo, se propone resolver instancias de diferentes tamaños del caso de estudio. Dado que, tanto el esfuerzo computacional, como la calidad de la solución, dependen del número de bloques y sub-bloques considerados en la definición del problema, se proponen 10 instancias que derivan del ejemplo original, los cuales varían en la cantidad de productos a procesar en el astillero y el tipo de almacenamiento intermedio, adoptando la política NIS o UIS. Notar que, como se manifestó en las secciones anteriores, el objetivo de optimización es la minimización del makespan para todas las instancias de problemas. Las características principales de todas las instancias resueltas se resumen en la Tabla 2.1. La expresión $N \times M$ se refiere a una embarcación construida con $N$ bloques y $M$ sub-bloques. La cantidad de productos ($NxM$) que se procesan está estrechamente relacionada con el tamaño del problema, lo que determina la complejidad combinatoria de los modelos.

Como se puede observar, el problema más pequeño abordado comprende 10 sub-bloques y 5 bloques, mientras que el más grande representa el caso de estudio real, en el cual el barco se construye a partir de 50 sub-bloques y 25 bloques. Notar que, se procesan

el doble de productos en las primeras etapas del sistema productivo que en las últimas. Si bien las instancias que involucran menos bloques no representan una situación real en un astillero, permiten evaluar la escalabilidad de los diferentes enfoques de optimización propuestos.

Tabla 2.1. Instancias del problema FFSP – Industria naval.

| Instancias del problema | Bloques x Sub-bloques $N \times M$ | Política de almacenamiento intermedio |
|---|---|---|
| P.01 | $5 \times 10$ | UIS |
| P.02 | $5 \times 10$ | NIS |
| P.03 | $10 \times 20$ | UIS |
| P.04 | $10 \times 20$ | NIS |
| P.05 | $15 \times 30$ | UIS |
| P.06 | $15 \times 30$ | NIS |
| P.07 | $20 \times 40$ | UIS |
| P.08 | $20 \times 40$ | NIS |
| P.09 | $25 \times 50$ | UIS |
| P.10 | $25 \times 50$ | NIS |

## 2.4.2. Discusión de resultados

Para cada una de las instancias definidas en la Tabla 2.1, se determina el programa óptimo de operaciones a partir de las formulaciones MILP introducidas en la sección 2.3 del presente capítulo. En primer lugar, se realizan ejecuciones de cada instancia para evaluar el desempeño de la formulación basada en el concepto de "time-slots", utilizando las 3 alternativas planteadas, y descriptas en la sección 2.3.2, para el cálculo del número de slots postulados en cada unidad de procesamiento $k$ (equivalente a un total de 30 ejecuciones): (i) suponiendo el número máximo de slots necesarios para procesar todas las órdenes en cada unidad ($BS1$); (ii) suponiendo que al menos un slot es utilizado en

todas las unidades que pertenecen a la misma etapa ($BS2$); y (iii) suponiendo la expresión propuesta por Fumero et al. (2012), en la cual se tiene como hipótesis que las unidades correspondientes a una misma etapa se usan en orden ascendente ($BS3$). La Tabla 2.2 resume las estadísticas computacionales obtenidas para cada uno de los ejemplos, detallando la mejor solución encontrada, el valor de GAP y los requerimientos computacionales asociados; mientras que, la Tabla 2.3 refleja el tamaño de los problemas, en términos de variables binarias y continuas y restricciones lineales. Notar que, como se ha mencionado, el tiempo límite predefinido, en principio, es de 1 hora de tiempo de CPU (3600s) para todos los ejemplos.

Tabla 2.2. Estadísticas computacionales de las formulaciones basadas en "time-slots".

| Inst. | Modelo $BS1$ | | | Modelo $BS2$ | | | Modelo $BS3$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Solución MILP | GAP (%) | Tiempo CPU (s) | Solución MILP | GAP (%) | Tiempo CPU (s) | Solución MILP | GAP (%) | Tiempo CPU (s) |
| P.01 | 126,3 | 0 | 8,1 | 126,3 | 0 | 1,0 | 126,3 | 0 | 0,4 |
| P.02 | 126,3 | 0 | 25,7 | 126,3 | 0 | 2,0 | 126,3 | 0 | 0,8 |
| P.03 | 160,4 | 12,8 | 3600 | 160,8 | 13,1 | 3600 | 160,7 | 13,1 | 3600 |
| P.04 | 162,1 | 13,9 | 3600 | 160,4 | 13,1 | 3600 | 162,8 | 13,1 | 3600 |
| P.05 | 224,2 | 31,6 | 3600 | 212,3 | 27,8 | 3600 | 207,0 | 25,9 | 3600 |
| P.06 | 239,6 | 36,0 | 3600 | 214,5 | 28,5 | 3600 | 211,5 | 27,5 | 3600 |
| P.07 | 249,8 | 33,2 | 3600 | 247,9 | 32,7 | 3600 | 231,1 | 27,8 | 3600 |
| P.08 | 257,5 | 35,2 | 3600 | 245,8 | 32,1 | 3600 | 240,4 | 30,6 | 3600 |
| P.09 | 323,8 | 48,1 | 3600 | 315,7 | 42,9 | 3600 | 299,8 | 40,0 | 3600 |
| P.10 | 332,9 | 49,5 | 3600 | 325,5 | 48,4 | 3600 | 321,4 | 47,7 | 3600 |

A partir de las Tablas 2.2 y 2.3, se puede observar que para problemas de tamaño reducido, los tres modelos basados en "time-slots" reportan la solución óptima en pocos segundos de tiempo de CPU. Por otro lado, para instancias mayores, se aprecia una

reducción drástica en el tamaño del modelo a medida que se aplican expresiones más sofisticadas para el cálculo del número slots postulados. Por ejemplo, utilizando el modelo *BS*2 en lugar del modelo *BS*1, para resolver el problema de mayor tamaño (P.10), se logra reducir un 36% la cantidad de restricciones, 38% el número de variables binarias y 32% el de variables continuas; mientras que aplicando *BS*3 se logra una reducción aún mayor, del 53%, 54% y 51% más, respectivamente. Sin embargo, los diferentes modelos no consiguen encontrar la solución óptima para aquellas instancias mayores a 10 bloques y 20 sub-bloques (P.03), es decir, para problemas de tamaño mediano y grande. Además, la reducción planteada en los tamaños de los modelos no se ve reflejada en una ayuda para la convergencia de los mismos, ya que el valor de GAP se sigue manteniendo en un nivel alto (mayor al 40% para casos de tamaño real, como P.09 y P.10).

Tabla 2.3. Tamaño de los modelos basados en "time-slots".

| Inst. | Modelo *BS*1 | | | Modelo *BS*2 | | | Modelo *BS*3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | Var. Bin. | Var. Cont. | Restr. | Var. Bin. | Var. Cont. | Restr. | Var. Bin. | Var. Cont. | Restr. |
| P.01 | 2475 | 2887 | 10767 | 625 | 909 | 2921 | 485 | 811 | 2431 |
| P.02 | 2475 | 2947 | 19927 | 625 | 969 | 5351 | 485 | 871 | 4441 |
| P.03 | 9900 | 10022 | 40062 | 4580 | 4862 | 35376 | 2130 | 2712 | 9476 |
| P.04 | 9900 | 10142 | 76662 | 4580 | 4862 | 35376 | 2130 | 2832 | 17876 |
| P.05 | 22275 | 21407 | 87772 | 14295 | 13817 | 56366 | 5550 | 6535 | 23971 |
| P.06 | 22275 | 21587 | 170122 | 14295 | 13997 | 109016 | 5550 | 6715 | 45931 |
| P.07 | 39600 | 37042 | 153907 | 28960 | 27142 | 112581 | 10720 | 12026 | 45331 |
| P.08 | 39600 | 37282 | 300307 | 28960 | 27382 | 219381 | 10720 | 12266 | 87571 |
| P.09 | 61875 | 56927 | 238467 | 48575 | 47717 | 187221 | 17375 | 19091 | 72831 |
| P.10 | 61875 | 57227 | 467217 | 38075 | 38822 | 301317 | 17375 | 18866 | 140277 |

A continuación, se selecciona el modelo de "time-slots" *BS*3 para ser comparado con las otras dos formulaciones presentadas en este capítulo, a fin de poder comparar el rendimiento de los tres enfoques puros de optimización. Las estadísticas computacionales

obtenidas para cada una de las corridas realizadas y los correspondientes tamaños de los modelos, se resumen en la Tabla 2.4 y Tabla 2.5, respectivamente.

Tabla 2.4. Estadísticas computacionales de los tres enfoques alternativos de optimización.

| Inst. | Formulación basada en "time-slots" - $BS3$ | | | Formulación basada en precedencia inmediata | | | Formulación basada en precedencia general | | |
|---|---|---|---|---|---|---|---|---|---|
| | Solución MILP | GAP (%) | Tiempo CPU (s) | Solución MILP | GAP (%) | Tiempo CPU (s) | Solución MILP | GAP (%) | Tiempo CPU (s) |
| P.01 | 126,3 | 0 | 0,4 | 126,3 | 0 | 2,1 | 126,3 | 0 | 2,3 |
| P.02 | 126,3 | 0 | 0,8 | 126,3 | 0 | 2,1 | 126,3 | 0 | 2,1 |
| P.03 | 160,7 | 13,1 | 3600 | 170,6 | 18,0 | 3600 | 160,1 | 12,7 | 3600 |
| P.04 | 162,8 | 13,1 | 3600 | 161,2 | 13,3 | 3600 | 161,4 | 13,4 | 3600 |
| P.05 | 207,0 | 25,9 | 3600 | 258,8 | 40,8 | 3600 | 202,4 | 24,3 | 3600 |
| P.06 | 211,5 | 27,5 | 3600 | NA* | - | 3600 | 210,6 | 27,2 | 3600 |
| P.07 | 231,1 | 27,8 | 3600 | 425,8 | 60,8 | 3600 | 229,2 | 27,2 | 3600 |
| P.08 | 240,4 | 30,6 | 3600 | NA* | - | 3600 | 240,2 | 30,6 | 3600 |
| P.09 | 299,8 | 40,0 | 3600 | 823,1 | 78,1 | 3600 | 290,8 | 37,9 | 3600 |
| P.10 | 321,4 | 47,7 | 3600 | NA* | - | 3600 | 319,1 | 47,4 | 3600 |

* No se reporta una solución factible dentro de 3600 segundos de tiempo de CPU.

Los resultados de ambas tablas permiten hacer un análisis comparativo entre los modelos alternativos y resaltar la alta complejidad combinatoria de las instancias analizadas cuando se resuelven con enfoques rigurosos de optimización. A partir de la Tabla 2.4, se puede deducir que los tres métodos MILP analizados resuelven rápidamente los pequeños problemas (P.01 y P.02), encontrando la solución óptima en unos pocos segundos de tiempo de CPU. Sin embargo, el requerimiento computacional crece drásticamente a medida que aumenta el tamaño del problema. Se puede observar que en los ejemplos restantes, P.03-P.10, los modelos rigurosos no pueden garantizar la optimalidad en un tiempo computacional aceptable, es decir, no pueden alcanzar la solución óptima después de 1 hora de tiempo de CPU. Estos resultados exhiben claramente

la debilidad de los métodos exactos para la resolución del problema de ensamblado de bloques, conduciendo rápidamente a un tamaño de modelo intratable con formulaciones matemáticas monolíticas.

Tabla 2.5. Tamaño de los tres modelos exactos basados en enfoques alternativos de optimización.

| Instancias | Formulación basada en "time-slot" - $BS3$ | | | Formulación basada en precedencia inmediata | | | Formulación basada en precedencia general | | |
|---|---|---|---|---|---|---|---|---|---|
| | Var. Bin. | Var. Cont. | Restr. | Var. Bin. | Var. Cont. | Restr. | Var. Bin. | Var. Cont. | Restr. |
| P.01 | 2475 | 2887 | 10767 | 2780 | 7397 | 21907 | 1390 | 102 | 2326 |
| P.02 | 2475 | 2947 | 19927 | 2780 | 7397 | 21907 | 1390 | 102 | 2326 |
| P.03 | 9900 | 10022 | 40062 | 10510 | 28742 | 86547 | 5255 | 202 | 9626 |
| P.04 | 9900 | 10142 | 76662 | 10510 | 28742 | 86547 | 5255 | 202 | 9626 |
| P.05 | 22275 | 21407 | 87772 | 23190 | 64037 | 193962 | 11595 | 302 | 21901 |
| P.06 | 22275 | 21587 | 170122 | 23190 | 64037 | 193962 | 11595 | 302 | 21901 |
| P.07 | 39600 | 37042 | 153907 | 40820 | 113282 | 344152 | 20410 | 402 | 39151 |
| P.08 | 39600 | 37282 | 300307 | 40820 | 113282 | 344152 | 20410 | 402 | 39151 |
| P.09 | 61875 | 56927 | 238467 | 63400 | 176477 | 537117 | 31700 | 502 | 61376 |
| P.10 | 61875 | 57227 | 467217 | 64650 | 176727 | 585867 | 31700 | 502 | 61376 |

Además, para los ejemplos que involucran 15 o más bloques y una política de almacenamiento NIS, el enfoque basado en precedencia inmediata no tiene la capacidad de encontrar una solución factible dentro de 3600s de tiempo computacional. En estos casos, el "solver" termina sin encontrar una solución factible debido al error de "capacidad de memoria excedida". En contraste, las otras formulaciones matemáticas pueden reportar soluciones factibles con cierto porcentaje de GAP en una 1 hora de tiempo de cómputo.

Si bien, cuando se consideran problemas de tamaño mediano a grande, tanto el modelo basado en precedencia general, como el basado en intervalos de tiempo, no pueden alcanzar la solución óptima dentro del tiempo predefinido, se puede notar que la

primera formulación matemática presenta un rendimiento computacional levemente mejor (de hasta un 3%).

Para el problema completo, que considera todos los bloques involucrados en la estrategia constructiva de la embarcación de escala industrial, y que corresponden a las instancias P.09 y P.10, según política de almacenamiento adoptada (UIS o NIS), ningún enfoque logra reportar una buena solución en 1 hora de tiempo de CPU. Tal es así que, aunque a través del modelo basado en el concepto de precedencia general global se puede reducir a la mitad la cantidad de variables binarias, todas las soluciones encontradas para estas instancias de gran tamaño se obtienen con un gran valor de GAP (por encima del 37%). Por ejemplo, la instancia P.09 con un almacenamiento intermedio ilimitado (política UIS) resulta en un modelo MILP, basado en precedencia general, con 61376 restricciones lineales, 31700 variables binarias y 502 variables continuas. La mejor solución factible reportada por el "solver" en 1 hora de procesamiento es de un $MK = 290{,}8$ días de makespan con un GAP de optimalidad del 37,9%. Mientras que, el tamaño del modelo basado en "time-slots" asciende a 238467 restricciones, 61875 variables binarias y 56927 variables continuas, alcanzando una solución de 299,8 días y un valor de GAP del 40 % para P.09. Para la instancia P.10, la mejor solución factible encontrada bajo el mismo concepto fue de un $MK = 319{,}1$ días, con un GAP de optimalidad del 47,4%, aplicando precedencia generalizada y 321,4 días, con un GAP del 47,7%, aplicando la estrategia de "time-slots".

La mejor solución encontrada para P.10, mencionada en el párrafo anterior, se muestra en la Figura 2.8 a través de un diagrama Gantt. En la gráfica se distinguen con el mismo color y número, los bloques y sus respectivos sub-bloques, y se delimitan las unidades en paralelo de cada etapa de procesamiento. De este modo, se puede apreciar la dimensión del problema en cuanto a la cantidad de unidades y de productos involucrados, sobre todo en las primeras etapas, las cuales se encuentran destinadas al procesamiento de una mayor cantidad de órdenes de productos (sub-bloques).

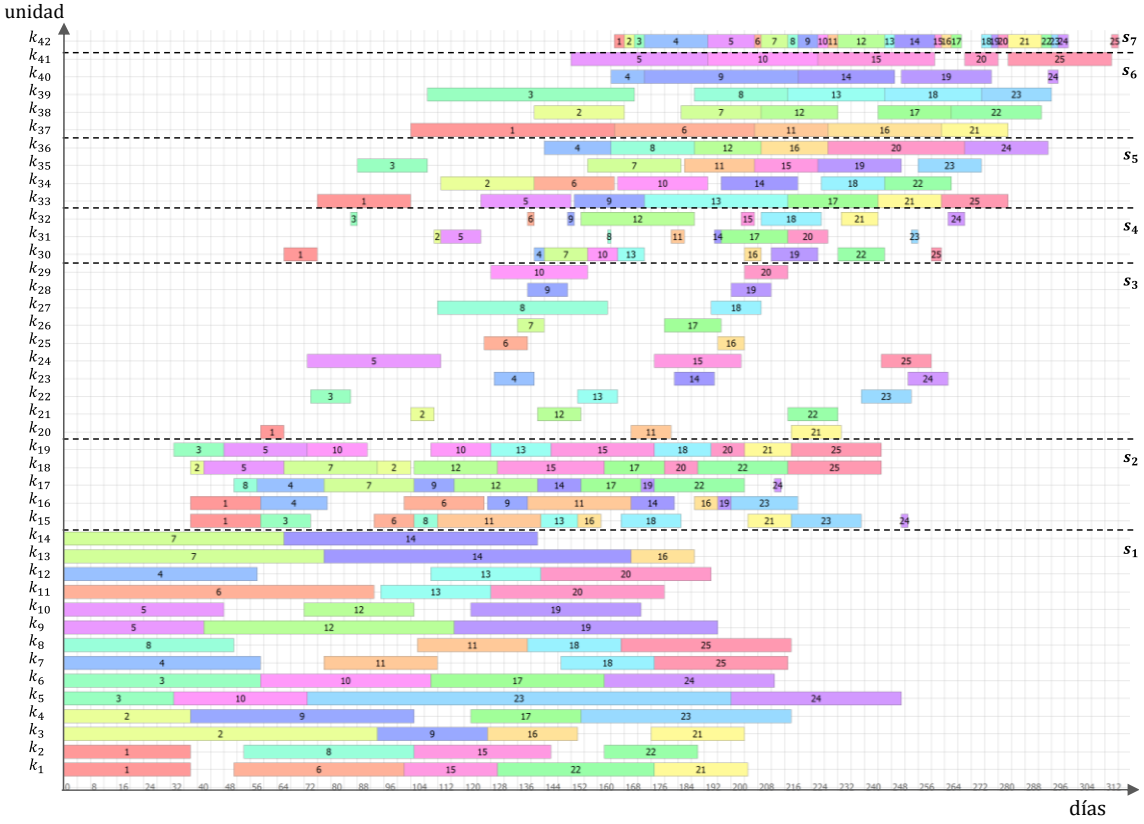



Figura 2.8. Diagrama Gantt de P.10 a partir del enfoque de precedencia general - $MK = 319{,}1$ días.

En resumen, a partir de los resultados obtenidos, se puede concluir que:

i. El costo computacional en los tres modelos de tiempo continuo crece exponencialmente con el tamaño del problema, debido a los varios miles de variables discretas y restricciones involucradas, característica típica de los problemas NP-hard. Tal es así que, sólo se reporta la solución óptima, dentro del tiempo computacional establecido (1 hora de tiempo de CPU), para las instancias de menor tamaño.

ii. El modelo basado en precedencia inmediata resulta ineficiente cuando se resuelven problemas de tamaño mediano o grande, no reportando, en algunos casos, una solución factible.

iii. Es notable la diferencia entre los tamaños de los enfoques de optimización para las diferentes instancias. Las formulaciones basadas en “time-slots” y precedencia inmediata,

como mínimo, duplican la cantidad de variables y restricciones con respecto a la formulación basada en precedencia general global.

iv. El enfoque basado en el concepto de precedencia general es el más eficiente para todos los tamaños del caso de estudio, con respecto a la solución reportada, cuando se alcanza el criterio de terminación de tiempo (3600 segundos de cómputo).

Es importante destacar que, al tratarse de un proceso de producción de larga duración (que involucra varios meses), se podría suponer que no es necesario disponer de una herramienta computacional a nivel operativo que brinde una buena solución en menos de una 1 hora de tiempo de CPU, sino que se permite una cierta flexibilidad en el tiempo necesario para obtener una buena solución. Razón por la cual, se realiza un análisis para determinar la evolución de la solución dejando al "solver" libre, sin restricción de tiempo computacional, es decir, únicamente con un valor de GAP del 0% como condición de terminación. La aplicación de dicho criterio a los modelos basados, tanto en precedencia general, como en "time-slots", da como resultado que el "solver" termina debido al error de "capacidad de memoria excedida", mencionado anteriormente, sin encontrar la solución óptima.

La mejor solución factible encontrada durante el tiempo de búsqueda, a partir de la formulación de "time-slots", es de 288,6 días con un GAP de optimalidad del 37,5%, la cual se refleja en una reducción del 10,2% con respecto a la solución reportada en 1 hora de tiempo de CPU. Mientras que, aplicando la formulación de precedencia general se obtiene una solución de 284,7 días con un 36,6% de valor de GAP y una reducción adicional del 10,8%. Esta última solución es representada a través del "schedule" mostrado en la Figura 2.9.

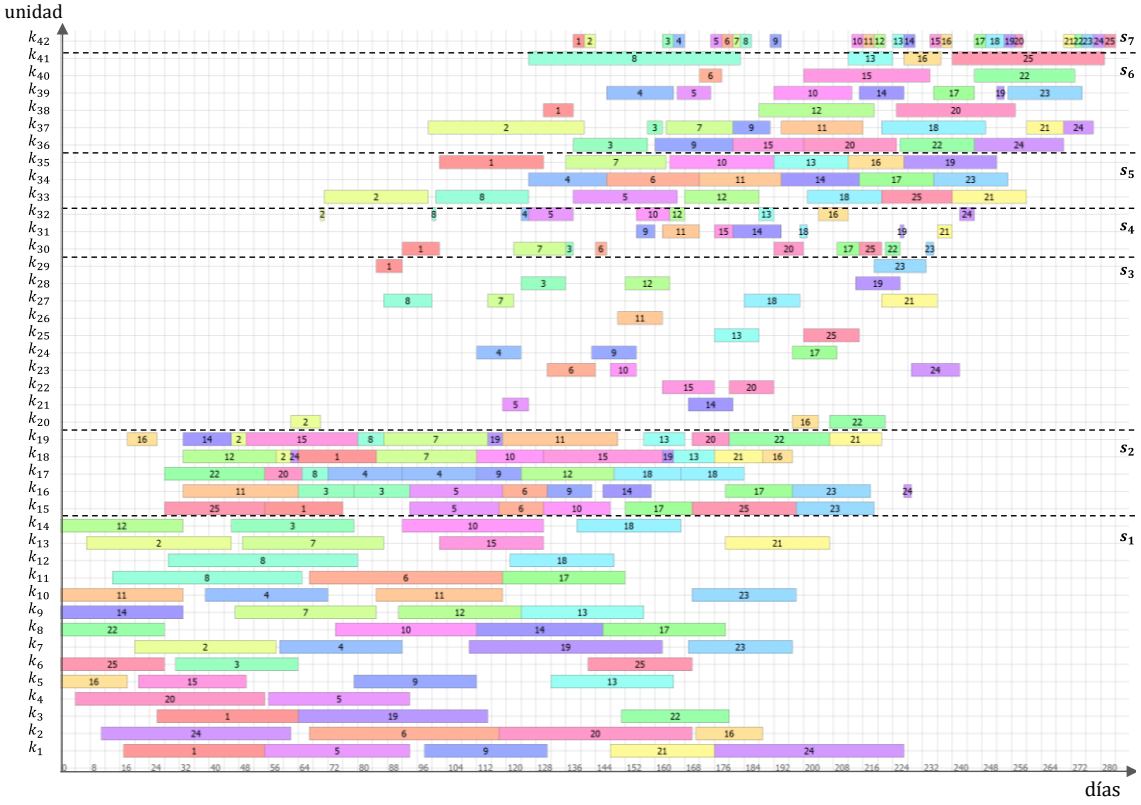



Figura 2.9. Diagrama Gantt de P.10 sin restricción de tiempo computacional - $MK = 284{,}7$ días.

## 2.5. Conclusiones

A lo largo de este capítulo se presentaron diferentes formulaciones de optimización MILP, con representación continua del tiempo, para resolver el problema de programación detallada de operaciones de un proceso flow shop flexible (FFSP) de escala industrial, presente en la industria naval. El problema planteado está basado en un caso de estudio real, que se da particularmente en el proceso de construcción de barcos con un diseño modular integrado, es decir, en la fabricación de embarcaciones de diferentes tamaños y diseños basada en el proceso llamado “ensamblado de bloques”.

Con el objetivo de evaluar el desempeño que presentan los diferentes enfoques exactos de optimización para resolver problemas de “scheduling” para procesos de manufactura de tamaño industrial con las características planteadas, se propusieron tres

modelos MILP de tiempo continuo basados en los siguientes enfoques: "time-slots", precedencia inmediata y precedencia general global. Para la resolución de dicho problema se introdujeron los cambios necesarios para modelar operaciones de ensamblado y secuencias de montaje específicas. Además, se definió un conjunto de instancias derivadas del mismo caso de estudio, el cual comprendía la fabricación y montaje de 50 sub-bloques y 25 bloques, en 7 etapas de procesamiento con 42 unidades en paralelo. Tales instancias, variaban en tamaño (cantidad de bloques y sub-bloques involucrados en el diseño modular) y en la política de almacenamiento intermedio adoptada. Para la resolución de cada una de ellas, se utilizaron las tres formulaciones planteadas de modo poder hacer un análisis comparativo detallado entre las mismas.

Los resultados reportados revelaron que, el enfoque de precedencia general global presentó una eficiencia levemente superior frente las formulaciones basadas en "time-slots" y en precedencia inmediata, para dar solución a todas las instancias. Sin embargo, para aquellas de tamaño mediano y grande, ninguna de las formulaciones pudo asegurar la optimalidad de la solución en menos de una hora de tiempo de CPU, reportando valores de GAP entre 25% y 47%. Sólo para problemas de tamaño reducido resultaron ser computacionalmente eficientes. Tal es así que, ninguna formulación pudo converger a la solución óptima del problema completo al no considerar límite de tiempo computacional. En todos los casos, el "solver" terminó por un error de "capacidad de memoria excedida". Por consiguiente, los modelos presentados sólo podrían ser utilizados para obtener una solución primaria de un problema de programación complejo del mundo real.

Dada la limitación inherente de estos métodos para alcanzar soluciones de buena calidad para problemas complejos de programación en un tiempo razonable, se debe recurrir a diferentes estrategias de solución. Razón por la cual, en el siguiente capítulo de la presente tesis se propone un procedimiento iterativo basado en los modelos MILP presentados. Se propone resolver una secuencia de modelos MILP de menor tamaño y por lo tanto, con costos computacionales inferiores, de manera de aprovechar la robustez de los enfoques exactos y las ventajas de los enfoques heurísticos.

# *Nomenclatura*

## *Conjuntos*

$I$ conjunto de órdenes de productos, correspondientes a bloques y sub-bloques (índice $i, i'$)

$K$ conjunto de unidades de procesamiento (índice $k, k'$)

$S$ conjunto de etapas de procesamiento (índice $s, s'$)

$P$ conjunto de slots (índice $p, p'$)

$I^b$ conjunto de bloques

$I^{sb}$ conjunto de sub-bloques

$SB_i$ conjunto de sub-bloques que conforman a cada bloque $i \in I^b$

$S^b$ conjunto de etapas de procesamiento $s$ disponibles para procesar los bloques $i \in I^b$

$S^{sb}$ conjunto de etapas de procesamiento $s$ disponibles para procesar los sub-bloques $i \in I^{sb}$

$S^a$ conjunto de etapas de ensamblado $s$ de sub-bloques $i \in I^{sb}$

$S^a_{seq}$ conjunto de etapa de ensamblado $s$ disponible para el montaje final de los bloques $i \in I^b$

$P_k$ conjunto de slots de la unidad de procesamiento $k$ (formulación MILP basada en "time-slots")

$K_s$ conjunto de unidades en paralelo de la etapa $s$

***Parámetros***

$tp_{is}$ tiempo de procesamiento de la orden $i$ en la etapa $s$

$Seq_i$ secuencia de montaje de los productos $i \in I^b$

$M$ constante big-M

***Variables Continuas***

$Ts_{is}$ tiempo de inicio de la orden de producto $i$ en la etapa de procesamiento $s$

$Tf_{is}$ tiempo de finalización de la orden de producto $i$ en la etapa de procesamiento $s$

$Ts_{pk}$ tiempo de inicio del slot $p$ en la unidad de procesamiento $k$ (formulación basada en "time-slots")

$Tf_{pk}$ tiempo de finalización del slot $p$ en la unidad de procesamiento $k$ (formulación basada en "time-slots")

$MK$ makespan

***Variables binarias***

$W_{ii'k}$ vale 1 si la orden de producto $i$ se procesa antes (formulación MILP basada en precedencia general), o inmediatamente antes (formulación de precedencia inmediata), que la orden $i'$ en la unidad de procesamiento $k$

$W_{ipk}$ vale 1 si la orden de producto $i$ se asigna en el slot $p$ de la unidad de procesamiento $k$ (formulación basada en "time-slots")

$Y_{ik}$ vale 1 si la orden de producto $i$ se procesa en la unidad de procesamiento $k$

$YF_{ik}$ vale 1 si la orden de producto $i$ es la primera en ser procesada en la unidad de procesamiento $k$

# Capítulo 3

# *Estrategia de descomposición basada en programación matemática para problemas de escala industrial*

## 3.1. Introducción

En el capítulo anterior, se presentaron tres formulaciones matemáticas alternativas, con manejo continuo del tiempo, para dar solución al problema de programación de operaciones en entornos FFSP que consideran operaciones de ensamblado de productos intermedios. Si bien dichos enfoques de optimización rigurosos fueron evaluados para distintas instancias de un caso de estudio real, proveniente de la industria naval, los resultados obtenidos reflejaron la debilidad que presentan los modelos matemáticos para la resolución de problemas de la vida real. De este modo, quedó demostrado que, el uso de técnicas monolíticas de optimización, como único enfoque de resolución, no resulta suficiente para abordar problemas de programación de operaciones de escala industrial.

Tal y como se lo ha mencionado anteriormente en esta tesis, dada la naturaleza combinatoria y el amplio conjunto de decisiones que se deben abordar en un problema de "scheduling" de escala industrial, los sistemas de manufactura son considerados problemas NP-hard (Pinedo, 2016). El esfuerzo computacional para encontrar una buena solución tiende a ser tan importante como el problema de programación en sí mismo; ya que la industria exige soluciones que sean óptimas, o al menos buenas soluciones subóptimas, y rápidas de alcanzar. En consecuencia, resulta necesario recurrir a otras metodologías de solución.

En la literatura, existen numerosas investigaciones que se han centrado en el desarrollo de técnicas heurísticas o metaheurísticas (França et al., 1996; Osman y Kelly, 1996; Pacciarelli, 2002; Pan y Ruiz, 2013; Ruiz y Maroto, 2005; Ruiz y Stützle, 2008) para la resolución eficiente de los problemas de programación de operaciones en entornos industriales. Por ejemplo, se han utilizado reglas de despacho, algoritmos genéticos, teoría de grafos, "tabu-search", métodos de optimización de colonias de hormigas, etc. Otros trabajos se han focalizado en la utilización de métodos híbridos. Algunas contribuciones destacadas en esta dirección se pueden encontrar en Castro et al., (2011), Chu et al. (2014), Zhang et al. (2016), entre muchos otros. Particularmente, el desarrollo de enfoques híbridos basados en simulación y optimización se convirtió en una estrategia de solución muy conveniente para mejorar los resultados obtenidos para los procesos de gran escala, por ejemplo, Basán et al. (2017) propusieron un algoritmo que integra un modelo matemático MILP y un modelo de simulación de eventos discretos para el proceso de construcción de barcos, considerando un sistema flow shop flexible (FFSP), como el abordado en el capítulo anterior. Si bien la herramienta presentada permite obtener programas de operaciones completos con tiempos computacionales razonables, las soluciones obtenidas no garantizan la optimalidad, ya que se aplican diversas heurísticas y no se considera el problema completo en la formulación matemática.

Por otro lado, se han reportado diferentes técnicas de descomposición-agregación y algoritmos de mejora para el problema de programación de operaciones en sistemas de

manufactura de múltiples etapas (Aguirre et al., 2012; Cóccola et al., 2015; Harjunkoski y Bauer, 2017; Kopanos et al., 2010; Méndez et al., 2000, 2006; Carlos A. Méndez y Cerdá, 2003; Roslöf et al., 2001) y enfoques de programación para la solución y el modelado con implementaciones industriales (Castro et al., 2018; Eberle et al., 2016; Harjunkoski et al., 2014b; Zeballos et al., 2011). Todas estas estrategias se basan generalmente en enfoques rigurosos de programación matemática, como los presentados en el Capítulo 2 (Basán et al., 2017a; Cerdá et al., 1997; Méndez et al., 2000, 2006; C. A. Méndez et al., 2001; Pinto y Grossmann, 1995b).

Con el fin de aprovechar los beneficios propios de las técnicas heurísticas y la robustez de los métodos exactos, en este capítulo se presenta una metodología de dos etapas basada en una estrategia de descomposición, para proporcionar una solución eficaz y de buena calidad a problemas complejos de programación de operaciones presentes en ámbitos industriales. Siguiendo la línea de investigación del capítulo anterior, se considera un ambiente productivo del tipo *flow shop flexible* (FFSP), con la particularidad de que algunas etapas del proceso llevan a cabo operaciones de ensamblado de productos intermedios. Cabe destacar que, el modelo matemático utilizado en la metodología iterativa está basado en el concepto de precedencia general global, el cual presentado en el capítulo anterior (sección 2.3.3).

La motivación principal de este capítulo es el desarrollo de un procedimiento de descomposición basado en etapas secuenciales que se resuelven de manera iterativa y que permiten mantener el número de variables utilizadas en un nivel razonable. La estrategia de solución propuesta es capaz de generar un programa completo de las operaciones para sistemas industriales de gran tamaño, requiriendo un esfuerzo computacional aceptable y además, de agilizar las decisiones de planificación con respecto a las asignaciones de recursos.

La aplicabilidad y la eficiencia de la estrategia de solución presentada se demuestran resolviendo las diferentes instancias del caso de estudio presentado en la

sección 2.4.1 sobre el proceso de construcción de barcos. El objetivo es encontrar el "schedule" que determine la mejor secuencia de montaje que minimice el tiempo total requerido para construir un barco a gran escala. Además, se presenta una comparación detallada entre los resultados alcanzados por una de las técnicas de resolución presentadas en el Capítulo 2 y los obtenidos a partir del enfoque iterativo propuesto. Adicionalmente, la estrategia de descomposición se utiliza para resolver una nueva variante, correspondiente a una configuración diferente de un astillero y el montaje de un buque basado en otra estrategia de construcción.

El capítulo se organiza de la siguiente manera. En la sección 3.2 se introduce la definición del problema exponiendo las principales hipótesis de trabajo. Luego, en la sección 3.3 se explica detalladamente el procedimiento de descomposición para la resolución iterativa de la formulación MILP. En la sección 3.4 se evalúa la performance computacional de la estrategia de solución propuesta y se realiza un análisis comparativo con los resultados obtenidos por los enfoques de optimización presentados en el anterior capítulo de la tesis. Finalmente, las conclusiones de este capítulo se discuten en la sección 3.6, y luego, se presenta la nomenclatura utilizada.

## 3.2. Descripción del problema

Para la aplicación de la estrategia de solución propuesta en este capítulo, se considera el proceso industrial de tipo flow shop flexible abordado en el Capítulo 2, donde además, como se ha mencionado, se pueden efectuar operaciones de ensamblado de productos intermedios para la obtención de productos finales de mayor valor agregado.

A modo de repaso, la Figura 3.1 muestra un ejemplo ilustrativo de un típico sistema FFSP secuencial con etapas de ensamblado, donde todos los productos siguen la misma receta de procesamiento. El ejemplo consiste en 5 órdenes de trabajo, equivalentes a las órdenes de sub-bloques consideradas en el capítulo anterior ($|I^{sb}|$), procesadas en 4 etapas productivas ($|S|$), con diferentes cantidades de unidades en paralelo $K_s$, para la

obtención de 2 productos finales ($|I^b|$): $i_6$ e $i_7$. Notar que, la segunda etapa $s_2$ de la línea lleva a cabo operaciones de montaje ($s_2 \in S^a$).

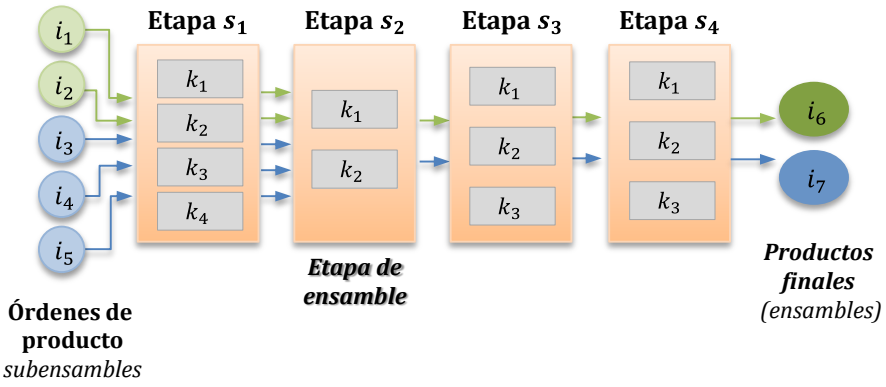



Figura 3.1. Sistema FFSP con operaciones de ensamblado.

Recordando las características principales del problema FFSP, en el sistema se pueden procesar un conjunto de productos $i \in I$ ($i = 1,2,\dots,|I|$), de los cuales algunos de ellos pueden ser subensambles, o también llamados productos intermedios, ($i \in I^{sb}$), a partir de los cuales se pueden obtener los productos finales ($i \in I^b$), donde se cumple que $I^{sb} \cup I^b = I$. Por lo tanto, para cada producto $i \in I^b$ se define un subconjunto correspondiente a sus subensambles $i' \in SB_i$. Asimismo, cada etapa $s \in S$ ($s = 1,2,\dots,|S|$) puede consistir en una o más unidades de procesamiento $k \in K_s$ ($k = 1,2,\dots,|K_s|$), donde $|K_s|$ es el número total de unidades en paralelo de la etapa $s$. Análogamente al capítulo anterior, se considera que existen etapas destinadas a efectuar actividades de montaje de productos intermedios, distinguiéndose dos tipos de etapas de ensamblado: $S^a$ y $S^a_{seq}$. Notar que, los productos no son procesados en todas las etapas $s \in S$, por lo tanto, se define un conjunto de aquellas etapas que procesan a cada orden $i$ como $S_i$. Asimismo, teniendo en cuenta que las unidades de cada etapa son idénticas, cada orden de producto $i$ se puede procesar en cualquier unidad de dicha etapa con el mismo tiempo de procesamiento $tp_{is}$.

Al igual que en el anterior capítulo, se contempla trabajar (i) sin restricciones de almacenamiento intermedio, suponiendo que existe capacidad ilimitada (UIS), así como la

situación opuesta, (ii) considerando que no existe almacenamiento intermedio (NIS). En cuanto a las políticas de espera entre etapas, se permite que un producto intermedio aguarde en la unidad que lo ha procesado, hasta que una unidad de la siguiente etapa esté disponible, el tiempo que sea necesario (UW), dando lugar a la política NIS/UW.

El criterio de performance considerado para alcanzar los objetivos es la minimización del tiempo total requerido para la obtención de los productos finales, es decir del makespan ($MK$), de manera de satisfacer todas las restricciones operativas y las secuencias de ensamblado preestablecidas, en el caso que las hubiera. Las decisiones en este nivel consisten en determinar el "scheduling" de producción (asignación, secuenciación y los tiempos de procesamiento de todas las órdenes de productos).

Cabe aclarar que, las hipótesis de trabajo y las decisiones de programación de las operaciones del problema considerado en el Capítulo 2 (sección 2.2.2) se mantienen en este capítulo para la nueva estrategia de solución propuesta. En resumen:

- Cada orden de producto sigue una secuencia preestablecida de procesamiento y en cada etapa es procesada por una única unidad.
- Existen etapas de procesamiento $s \in S^a$ y $s \in S^a_{seq}$ donde se realizan diferentes operaciones de ensamblado.
- Las etapas cuentan con unidades idénticas en paralelo, las cuales no pueden procesar más de una orden a la vez.
- Los productos finales $i \in I^b$ pueden estar formados por subensambles $i' \in SB_i$.
- Las unidades no pueden ser interrumpidas durante el procesamiento de una orden de producto.
- Existe disponibilidad ilimitada de materiales en la planta.
- Los parámetros del problema, tales como los tiempos de procesamiento ($tp_{is}$), son determinísticos y previamente definidos.

- No existen tiempos de transporte o transferencia del material.
- Los trabajos a procesar son independientes.
- Pueden existir restricciones de precedencia o de secuencia entre las órdenes de productos, por ejemplo, la secuencia predefinida de ensamblado de productos intermedios ($Seq_i$).
- Los tiempos de montaje resultan un factor clave para la estimación del rendimiento y la productividad de la planta.

## 3.3. Algoritmo de descomposición basado en una formulación matemática MILP

En un problema de programación de operaciones FFSP de tamaño real, suele existir una gran cantidad de productos que deben ser manipulados, procesados y ensamblados para obtener los resultados finales deseados. Esto involucra un importante número de decisiones discretas, lo que resulta en un problema de "scheduling" muy complejo, desde el punto de vista combinatorio. Razón por la cual, como se ha mencionado anteriormente, pertenece a la clase de problemas NP-hard, excediendo fácilmente las capacidades de resolución actualmente proporcionadas por los softwares de optimización disponibles. En consecuencia, y como quedó demostrado en el Capítulo 2, los modelos matemáticos en su forma pura (ver sección 2.3) presentan una eficiencia computacional que se deteriora rápidamente al aumentar el tamaño del problema, convirtiéndose en intratables por los "solvers" o generando soluciones pobres, muy lejanas al óptimo. Sin embargo, es posible combinar la robustez y rigurosidad de la programación matemática MILP con la flexibilidad de las reglas heurísticas, para desarrollar una estrategia de solución altamente eficiente.

La estrategia de solución propuesta radica principalmente en que, por lo general, la eficiencia computacional de los modelos MILP depende en gran medida de la cantidad de variables binarias utilizadas en el mismo. De este modo, en su estructura básica, el

procedimiento propuesto consiste en un algoritmo de descomposición que resuelve en cada iteración, versiones de tamaño reducido del modelo monolítico MILP. Esto da la posibilidad de mantener el número de variables utilizadas en un nivel aceptable, incluso para problemas de gran tamaño. La estrategia desarrollada, permite encontrar soluciones de buena calidad con muy poco esfuerzo computacional, lo cual se pondrá en manifiesto en los ejemplos analizados en la siguiente sección.

Es importante destacar que, a partir de los modelos MILP presentados en la sección 2.3.3 y evaluados posteriormente (ver resultados expuestos en la Tabla 2.4), se selecciona la formulación matemática basada en el concepto de precedencia general para el desarrollo de la metodología de descomposición, ya que de las tres alternativas propuestas, es la que mejor desempeño computacional presenta para todos los ejemplos considerados. Sin embargo, también es posible seleccionar cualquier de las dos formulaciones restantes como modelos centrales en la estrategia de solución propuesta, realizando modificaciones en el algoritmo.

La estructura general del algoritmo se muestra en la Figura 3.2. En una primera etapa, llamada *etapa constructiva* o *etapa de construcción*, se resuelve iterativamente el modelo MILP, insertando una orden de producto (y sus subensambles, en el caso que los tuviera) cada vez y fijando las variables de decisión en su valor óptimo. Una vez que todas las órdenes son incorporadas y programadas, el procedimiento obtiene un programa de operaciones completo y factible para el problema bajo estudio, dando por concluida la primera etapa del algoritmo.

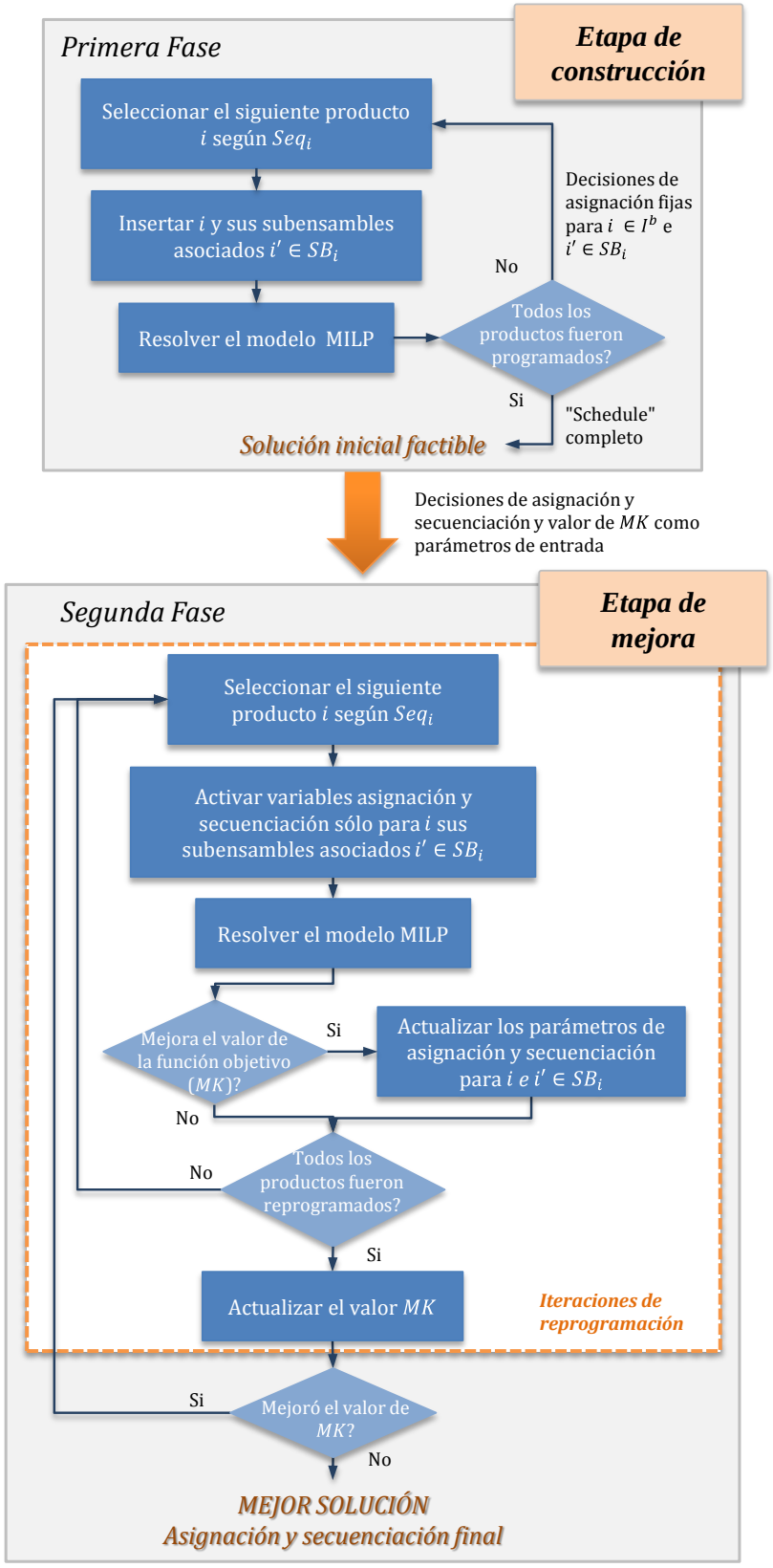



Figura 3.2. Esquema general del algoritmo iterativo basado en un modelo MILP para problemas FFSP.

Luego, una segunda etapa, llamada *etapa de mejora,* tiene como objetivo perfeccionar progresivamente, a través de iteraciones de reprogramación, la solución inicial reportada, con el fin de encontrar una solución de programación de alta calidad con un mínimo esfuerzo computacional. Cada iteración de mejora permite reprogramar una orden de producto por vez, por lo que se liberan las variables binarias asociadas a las decisiones de asignación y secuenciación de dicha orden. En el caso de que se incluya el ensamblado de productos en el problema de programación, se debe reprogramar a la vez las órdenes asociadas al producto y sus correspondientes subensambles.

De este modo, se reduce significativamente el espacio de búsqueda de optimización, permitiendo encontrar la solución óptima del modelo MILP que se resuelve en cada iteración. Si en alguna de las iteraciones se reportan mejoras en la función objetivo, es decir, la reprogramación permite reducir el makespan, se actualiza el valor $MK$ y se fijan las variables binarias de asignación y secuenciación. El algoritmo continúa hasta que no se encuentra una reasignación y/o reordenamiento de las órdenes que permita mejorar la función objetivo.

Notar que, en el algoritmo propuesto, se considera como función objetivo la minimización de makespan, sin embargo, se puede optar por funciones objetivos alternativas, como la minimización del costo total de operación, de la tardanza total de los trabajos, de la utilización de máquinas, entre otros.

Una vez finalizada la segunda etapa del algoritmo, se obtiene un programa de operaciones global correspondiente a la mejor solución encontrada para el problema FFSP bajo estudio. Los pseudocódigos de cada etapa se describen a continuación.

### 3.3.1. Primera fase: Etapa de construcción

La primera fase del algoritmo de descomposición tiene como objetivo generar una solución inicial que brinde un programa completo de operaciones con un bajo esfuerzo computacional. Por lo que, el problema de programación a gran escala se descompone de

manera iterativa en un subconjunto de órdenes de producto con el fin de reducir el espacio de búsqueda del "solver" y favorecer la resolución del problema.

El método constructivo empleado se basa en la técnica de inserción presentada por Kopanos et al. (2010) para resolver problemas de programación farmacéutica de gran escala. Adaptando la metodología propuesta por los autores al problema FFSP presentado en este capítulo, se obtiene una primera etapa donde se realiza un proceso iterativo de inserción y programación: en cada iteración se programa una cantidad predefinida de órdenes de productos hasta programar todas las órdenes involucradas en el problema. El programador puede definir, bajo el criterio que considere más conveniente según la naturaleza del problema, el número de órdenes de producto a insertar en cada iteración.

Es importante resaltar que, la cantidad de órdenes insertadas simultáneamente debe ser lo suficientemente pequeña para garantizar la resolución rápida del modelo MILP en cada iteración, y así generar un programa completo y factible en pocos segundos de tiempo de CPU. En este caso, se propone insertar (programar) una por una las órdenes de productos, ya que a partir de una serie de experimentos, se ha observado que la inserción de un mayor número de productos por iteración: (i) no garantiza encontrar una mejor solución en la fase de construcción y (ii) puede resultar en un mayor esfuerzo computacional.

Los productos de mayor valor agregado, es decir, las órdenes de productos ensamblados $i \in I^b$ se insertan de una en una en cada iteración, con sus correspondientes subensambles o partes $i' \in SB_i$, y se resuelve el modelo MILP de la sección 2.3.3, que asigna y secuencia las órdenes recientemente insertadas a las unidades de procesamiento, teniendo en cuenta aquellas ya programadas en anteriores iteraciones. El orden en que se insertan dichas órdenes puede variar según las características del problema (Roslöf et al., 2002, 2001). Por ejemplo, puede establecerse un orden lexicográfico o en función de una secuencia específica, como la predefinida a través del parámetro $Seq_i$ para el ensamblado que se lleve a cabo en alguna etapa $s \in S^a_{seq}$. Teniendo en cuenta el problema a resolver, se

propone insertar según la secuencia predefinida $Seq_i$, que a su vez, puede coincidir con el orden lexicográfico del conjunto $I^b$, como en el problema abordado en el Capítulo 2.

El pseudocódigo de la etapa de construcción se muestra en la Figura 3.3. El parámetro $iter$ identifica el número de órdenes de productos $i \in I^b$ que se inserta en la iteración actual, mientras que el parámetro booleano $active_i$ toma valor verdadero para todas las órdenes de producto $i$ consideradas en la ejecución del modelo MILP. Una vez que se inserta una nueva orden y las correspondientes a las partes asociadas, se resuelve el modelo MILP. Luego, se fijan los valores óptimos obtenidos de las variables binarias de asignación $Y_{ik}$ para órdenes de producto ya programadas. En otras palabras, las decisiones de asignación de órdenes a unidades se mantienen fijas en las siguientes iteraciones sin poder ser modificadas. Sin embargo, las decisiones de tiempo pueden cambiar.

Las decisiones de secuenciación se ignoran en esta fase del algoritmo, y se toma como hipótesis que la secuencia de producción de cada etapa de procesamiento sigue el orden establecido por la secuencia $Seq_i$, es decir, $W_{ii'k} = 1$ cuando $Seq_i < Seq_{i'}$. Estas suposiciones permiten reducir el espacio de búsqueda y acelerar la resolución del modelo en cada iteración. Por lo tanto, las restricciones (2.30) y (2.31) son reemplazadas por la siguiente:

$$Ts_{is} \geq Tf_{i's} - M(2 - Y_{ik} - Y_{i'k}) \tag{3.1}$$

$$\forall \Big( \big( (i,i') \in I^{sb} \big) \vee \big( (i,i') \in I^{b} \big) \Big), s \in S_i, k \in K_s : Seq_i < Seq_{i'}$$

El procedimiento de la etapa constructiva termina cuando se han programado todas las órdenes de productos (ensambles y sus correspondientes subensambles). La solución reportada $MK$ en esta fase correspondiente al programa de operaciones completo obtenido, se guarda en el parámetro $Sol_{inicial}$. Los valores de las variables binarias de decisión, $Y_{ik}$ y $W_{ii'k}$, se almacenan en los parámetros $sY_{ik}$ y $sW_{ii'k}$, respectivamente, para su posterior utilización en la etapa de mejora.

Set $iter = 1, active_i = false$
WHILE $iter = Seq_i$ and $i \in I^b$
    LOOP ($i \in I^b$ *and* $iter = Seq_i$ )
        $active_i = true$
        LOOP ($i' \in SB_i$)
            $active_{i'} = true$
        END LOOP
    END LOOP
    Resolver Modelo MILP
    LOOP ($i \in I^b$ and $ord(i) = iter$ and $i' \in SB_i$)
        LOOP ($s \in S_i$ *or* $s \in S_{i'}$ )
            LOOP ($k \in K_s$)
                Fijar variables binarias $Y_{ik}$ y $Y_{i'k}$
            END LOOP
        END LOOP
    END LOOP
    $iter = iter + 1$
END WHILE
$CurrentSol = MK$
$sY_{ik} = Y_{ik}$
LOOP $((i, i') \in I^b$ and $Seq_i < Seq_{i'})$
    LOOP( $s \in S_i$)
        LOOP ($k \in K_s$)
            $sW_{ii'k} = 1$
        END LOOP
    END LOOP
END LOOP
LOOP $((i, i') \in I^{sb}$ and $Seq_i < Seq_{i'}))$
    LOOP ( $s \in S_i$)
        LOOP($k \in K_s$
            $sW_{ii'k} = 1$
        END LOOP
    END LOOP
END LOOP

Figura 3.3. Pseudocódigo de la etapa constructiva.

### 3.3.2. Segunda fase: Etapa de mejora

Las decisiones de asignación $sY_{ik}$ y secuenciación $sW_{ii'k}$, correspondientes al programa inicial factible proporcionado por la etapa constructiva ($Sol_{inicial}$), se consideran el punto de partida para la evaluación de posibles decisiones de reasignación y reordenamiento de las órdenes de productos, con el objetivo de mejorar sistemáticamente la solución del problema. En otras palabras, la fase de mejora se basa en un procedimiento iterativo de ciclo cerrado, que consiste en la reinserción secuencial de órdenes hasta no observar mejora alguna en el "schedule", es decir, en el valor de la función objetivo del modelo MILP (en este caso $MK$).

Una reprogramación completa de órdenes de producto (de manera simultánea), resulta poco práctica debido a la gran complejidad combinatoria asociada a la gran cantidad de órdenes involucradas, lo cual implica resolver el modelo MILP considerando el espacio completo de búsqueda. Sin embargo, optar por una alternativa de reasignación y reordenamiento de operaciones limitado, generalmente, puede mejorar la programación actual con un esfuerzo computacional relativamente bajo. En este sentido, y dado que el objetivo de este capítulo es proponer un algoritmo estándar general para problemas industriales FFSP a gran escala con un eficiente rendimiento computacional, se adopta una estrategia con el mínimo número de órdenes de reinserción, es decir, una a la vez, favoreciendo los tiempos de cómputo. De este modo, se dejan libres las decisiones de asignación de órdenes y secuenciación en unidades, para cada orden de producto ensamblado y sus partes asociadas, es decir, se liberan individualmente cada orden de producto $i \in I^b$ con sus partes $i' \in SB_i$, para que puedan ser reasignadas y ordenadas a otras unidades en paralelo en cada etapa, o reordenadas en la misma unidad. Notar que, el número de iteraciones (reinserciones) en cada bucle, será igual a la cantidad de órdenes de productos ensamblados: $|I^b|$.

En la Figura 3.4 se muestra el pseudocódigo desarrollado para modelar la etapa de mejora. Como se puede observar, cuando se inicia esta fase el parámetro $CurrentSol$

guarda la mejor solución encontrada en la etapa constructiva $Sol_{inicial}$ (que corresponde a la solución inicial factible encontrada), mientras que el parámetro $BestSol$ se inicializa con un valor alto. Luego, el procedimiento de mejora sistemática comienza con la primera ejecución, la cual implica varias iteraciones de reprogramación, tantas como órdenes de productos $i \in I^b$ involucre el proceso productivo abordado. En cada ejecución de esta etapa, el parámetro booleano $active_i$ toma valor verdadero para el subconjunto de órdenes que el algoritmo deje libre para su reprogramación: ensamble $i \in I^b$ y sus correspondientes partes $i' \in SB_i$.

Por lo tanto, la formulación MILP ejecutada en cada iteración, que tiene como restricciones (1) - (8), sólo tiene activas las variables binarias $Y_{ik}$ y $W_{ii'k}$ asociadas a las órdenes de productos $i$ con $active_i = verdadero$. Para las órdenes restantes, sólo se pueden tomar decisiones con respecto a los tiempos iniciales y finales de procesamiento en cada etapa ($Ts_{is}$ y $Tf_{is}$). De esta manera, la reasignación a otras unidades no está permitida para órdenes $i \in I^b$ e $i' \in SB_i$ con el parámetro $active_i$ establecido en $falso$. Del mismo modo, las decisiones de secuenciación de procesamiento de dichas órdenes (no reinsertadas) en las unidades $k$, son fijas, es decir, permanecen sin cambios. En otras palabras, algunas órdenes de productos se extraen del programa actual y se vuelven a insertar para mejorar el programa real. Cada vez que se ejecuta una acción de reprogramación, se actualiza la mejor solución encontrada para el problema en estudio.

Esta estrategia de descomposición permite reducir el número de variables binarias de la formulación matemática con respecto al enfoque de espacio completo, reduciendo drásticamente el tiempo de CPU necesario para resolver el modelo. Notar que, resolver el enfoque de espacio completo es equivalente a establecer $active_i = true \; \forall i \in I$.

Una vez que se aplica el paso de reprogramación para todas las órdenes de productos $i \in I^b$, el procedimiento verifica la mejora alcanzada en el valor del makespan. Si la última solución reportada ($CurrentSol$) es mejor que aquella encontrada hasta ese momento y reportada como $BestSol$, es decir, se cumple que $CurrentSol < BestSol$, el

algoritmo actualiza su valor a $BestSol = CurrentSol$, y ejecuta una nueva repetición de la fase de mejora. De lo contrario, el procedimiento iterativo finaliza y la última solución encontrada representa la mejor solución para el problema en cuestión.

---

Set $CurrentSol = Sol_{inicial}$
Set $BestSol = M$
WHILE $CurrentSol < BestSol$
    $BestSol = CurrentSol$
    $iter = 1$
    WHILE $iter = Seq_i \; and \; i \in I^b$
        $active_i = false$
        LOOP ($i \in I^b$ and $iter = Seq_i$)
            $active_i = true$
            LOOP ($i' \in I^{sb}$)
                $active_i = true$
            END LOOP
        END LOOP
        Resolver Modelo MILP
        $CurrentSol = MK$
        LOOP ($i \in I^b$ and $active_i = true$)
            $sY_{ik} = Y_{ik}$
            LOOP ($i' \in I^{sb}$)
                $sY_{i'k} = Y_{i'k}$
            END LOOP
        END LOOP
        LOOP ($((i, i') \in I^b \; or \; (i, i') \in I^b$) and
            ($active_i = true$ or $active_{i'} = true$))
            LOOP( $s \in S_i$)
                LOOP($k \in K_s$)
                    $sW_{ii'k} = W_{ii'k}$
                END LOOP
            END LOOP
        END LOOP
    $iter = iter + 1$
    END WHILE
END WHILE

---

Figura 3.4. Pseudocódigo de la etapa de mejora.

## 3.4. Aplicación del algoritmo iterativo y discusión de resultados

La aplicabilidad y potencialidad de la propuesta iterativa de resolución son evaluadas a través de la resolución de las instancias definidas para el caso de estudio real, presentadas en el Capítulo 2 de la presente tesis, y de una variante focalizada en una configuración diferente del mismo tipo de sistema productivo. Por lo tanto, se testean y analizan dos ejemplos basados en casos de estudio sobre compañías navales dedicadas a la construcción de grandes embarcaciones. Es importante mencionar que, sólo se exponen las configuraciones de los sistemas productivos y la cantidad de órdenes de productos a procesar en cada etapa, ya que por razones de confidencialidad no se pueden publicar los tiempos de procesamiento con los que trabajan los astilleros bajo estudio.

Asimismo, se lleva a cabo un análisis comparativo a partir de los resultados obtenidos por la estrategia de solución y los enfoques exactos, expuestos en la sección 2.4.2 de esta tesis. Particularmente, se tienen en cuenta aquellos resultados reportados por la formulación MILP basada en el concepto de precedencia general global, ya que es el enfoque que ha presentado cierta superioridad, en cuanto a eficiencia, frente a las demás formulaciones matemáticas.

El algoritmo iterativo es codificado en el lenguaje de modelado GAMS 24.9.2 y ejecutado en una computadora con un procesador Intel Xeon X5650 de cuatro núcleos (2,6 GHz) y 32 GB de RAM. Las formulaciones MILP, correspondientes a cada iteración del procedimiento, se resuelven con el optimizador CPLEX 12.7.1.0. Tal y como se ha implementado para el análisis de rendimiento de las formulaciones MILP del capítulo anterior, se impone un tiempo límite de 1 hora de CPU a cada corrida del modelo matemático utilizado como base en la estrategia de solución desarrollada.

### 3.4.1. Ejemplo 3.1

El primer caso de estudio abordado, como se ha mencionado, es el considerado y evaluado en el Capítulo 2. Por lo tanto, todos los datos del problema utilizados en este

ejemplo son similares a aquellos ya presentados en dicho capítulo. En resumen, el sistema productivo consiste en un astillero conformado por 7 etapas de procesamiento, de las cuales 2 realizan operaciones de ensamblado, y 42 unidades en total. En la Figura 3.5 se pueden observar las etapas de ensamblado: $s_3$ y $s_7$. La primera de ellas tiene como objetivo obtener bloques, considerados enmsables $i \in I^b$, a partir de la unión de sub-bloques, considerados partes o subensambles $i' \in SB_i$ de los anteriores. La segunda etapa de ensamblado, correspondiente a la última etapa del proceso productivo, lleva a cabo el montaje del barco siguiendo una secuencia predefina de ensamblado en bloques.

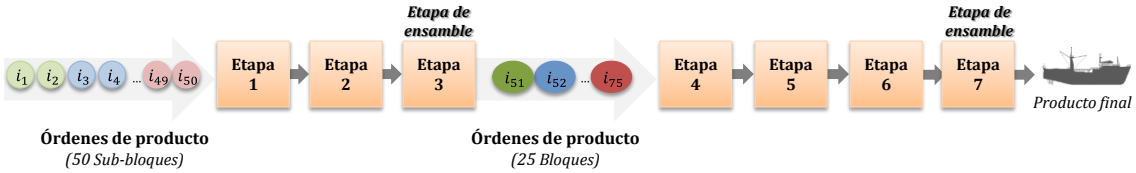



Figura 3.5. Estructura del proceso productivo – Ejemplo 3.1.

Tanto la cantidad de bloques y sub-bloques, como la secuencia de montaje final que se lleva a cabo en el dique seco, depende del diseño modular de cada barco. En particular, el barco que se plantea construir, como se definió en la sección 2.4.1, está formado por 25 bloques y 50 sub-bloques en total, considerando que cada bloque está formado por 2 sub-bloques de iguales características.

Luego de haber analizado los resultados obtenidos por las formulaciones MILP presentadas en el capítulo anterior, se resuelven las mismas instancias del caso de estudio con el procedimiento iterativo, con el objetivo de analizar su comportamiento y rendimiento computacional. De este modo, no sólo se valida la metodología desarrollada, sino también se comparan las fortalezas y limitaciones del nuevo enfoque frente a los modelos de optimización monolíticos.

En cada iteración del procedimiento, se realizan múltiples ejecuciones del modelo MILP. Si bien, en principio se definió un criterio de terminación de 1 hora de tiempo de

CPU, el gran desempeño que refleja la estrategia de solución, permite establecer un GAP de optimalidad del 0% como único criterio.

En la Tabla 3.1 se detallan los resultados computacionales reportados por el algoritmo para las 10 instancias presentadas en la Tabla 2.1 en la sección 2.4.1: (i) valor del makespan ($MK$) de la solución inicial obtenida en la primera etapa (*etapa de construcción*), (ii) mejor solución alcanzada en la segunda etapa (*etapa de mejora*) y (iii) requerimiento computacional total empleado. La última columna de dicha tabla contiene la información correspondiente a la cantidad de iteraciones ejecutadas por la etapa de mejora. Se puede notar que este número aumenta gradualmente conforme al tamaño del problema, llegando al caso del problema de tamaño industrial (P.10), donde el procedimiento converge a la mejor solución luego de 15 iteraciones. La Figura 3.6 ilustra este comportamiento para las instancias de mayor tamaño (P.07-P10), las cuales requieren un mayor número de iteraciones para converger a la mejor solución. Como se puede observar, la solución de la fase constructiva siempre es mejorada en la segunda etapa, por lo que se puede deducir que el algoritmo es capaz de generar rápidamente las soluciones factibles, para luego mejorarlas sistemáticamente con un mínimo esfuerzo computacional. Es importante recalcar que para todos los problemas, el algoritmo emplea menos de 60 segundos de tiempo de CPU para encontrar una solución cercana al óptimo.

Se compara la resolución del modelo completo (ver sección 2.4.2) con la resolución iterativa del mismo modelo, pero con espacios de búsqueda reducidos (Tabla 3.1). En la Tabla 3.2 se resumen las estadísticas computacionales del análisis comparativo. Las últimas dos columnas de la tabla calculan, en términos de porcentaje, la diferencia entre la mejor solución encontrada por el algoritmo y la solución reportada por el modelo MILP.

En la Tabla 3.2, en primer lugar, se puede observar que ambos enfoques reportan la misma solución para instancias de tamaño reducido. Esto significa que la estrategia de solución encuentra la solución óptima, al igual que el modelo MILP monolítico, para un problema de tamaño pequeño. De este modo, las soluciones obtenidas permiten la

validación de la metodología propuesta. En la Figura 3.7 se muestra el programa óptimo de operaciones para el problema de tamaño reducido P.02 (5 bloques, 10 sub-bloques y política de NIS), a través de un diagrama Gantt.

Tabla 3.1. Soluciones reportadas por el algoritmo en la etapa de construcción y mejora – Ejemplo 3.1.

| Instancias | Etapa de construcción | | Etapa de mejora | | | Porcentaje de mejora (%) |
|---|---|---|---|---|---|---|
| | Solución inicial | Tiempo CPU (s) | Mejor Solución | Tiempo total CPU (s) | Iteraciones | |
| P.01 | 144.0 | 0.6 | 126,3 | 2.3 | 3 | 12,3 |
| P.02 | 144.4 | 0.7 | 126,3 | 2.1 | 3 | 12,5 |
| P.03 | 176.3 | 1.5 | 160,0 | 11.1 | 6 | 9,2 |
| P.04 | 177.9 | 1.5 | 160,3 | 6.8 | 4 | 9,9 |
| P.05 | 239.1 | 2.1 | 200,2 | 16.3 | 6 | 16,3 |
| P.06 | 241.9 | 1.9 | 202,7 | 24.1 | 10 | 16,2 |
| P.07 | 248.8 | 2.8 | 221,0 | 37.6 | 15 | 11,2 |
| P.08 | 255.4 | 3.1 | 228,6 | 41.2 | 13 | 10,5 |
| P.09 | 301.1 | 4.6 | 262,8 | 49.8 | 11 | 12,7 |
| P.10 | 298.3 | 4.3 | 270,5 | 56.6 | 15 | 9,3 |

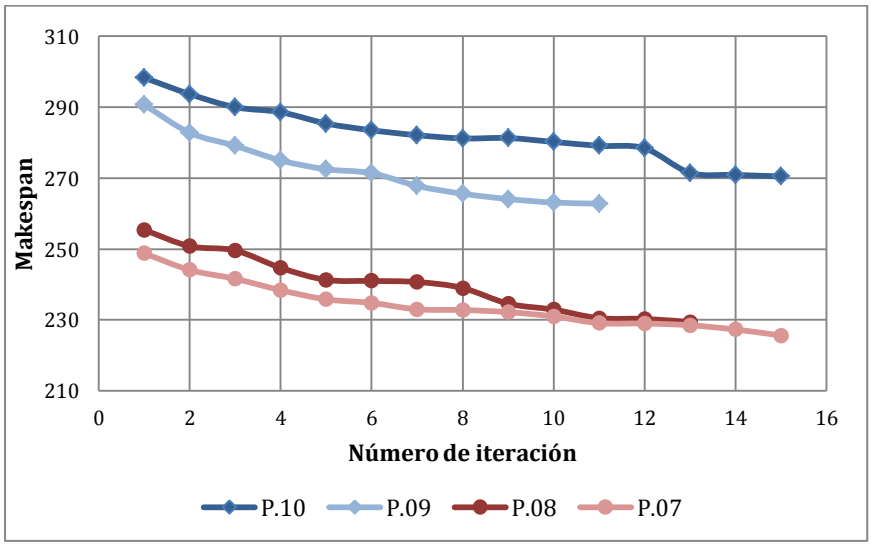



Figura 3.6. Evolución del makespan – Ejemplo 3.1.

Tabla 3.2. Comparación entre el modelo matemático y el método iterativo – Ejemplo 3.1.

| Instancias | Formulación basada en precedencia general | | | Estrategia basada en MILP | | Porcentaje de mejora en la solución (%) |
|---|---|---|---|---|---|---|
| | Solución MILP | GAP | CPU time (s) | Mejor solución | Tiempo total CPU (s) | |
| P.01 | 126,3 | 0 | 2,3 | 126,3 | 2,3 | 0 |
| P.02 | 126,3 | 0 | 2,2 | 126,3 | 2,1 | 0 |
| P.03 | 160,1 | 12,7 | 3600 | 160,0 | 11,1 | 0,1 |
| P.04 | 161,4 | 13,4 | 3600 | 160,3 | 6,8 | 0,7 |
| P.05 | 202,4 | 24,3 | 3600 | 200,2 | 16,3 | 1,1 |
| P.06 | 210,6 | 27,2 | 3600 | 202,7 | 24,1 | 3,8 |
| P.07 | 229,2 | 27,2 | 3600 | 221,0 | 37,6 | 3,6 |
| P.08 | 240,2 | 30,6 | 3600 | 228,6 | 41,2 | 4,8 |
| P.09 | 290,8 | 37,9 | 3600 | 262,8 | 49,8 | 9,6 |
| P.10 | 319,1 | 47,4 | 3600 | 270,5 | 56,6 | 15,2 |

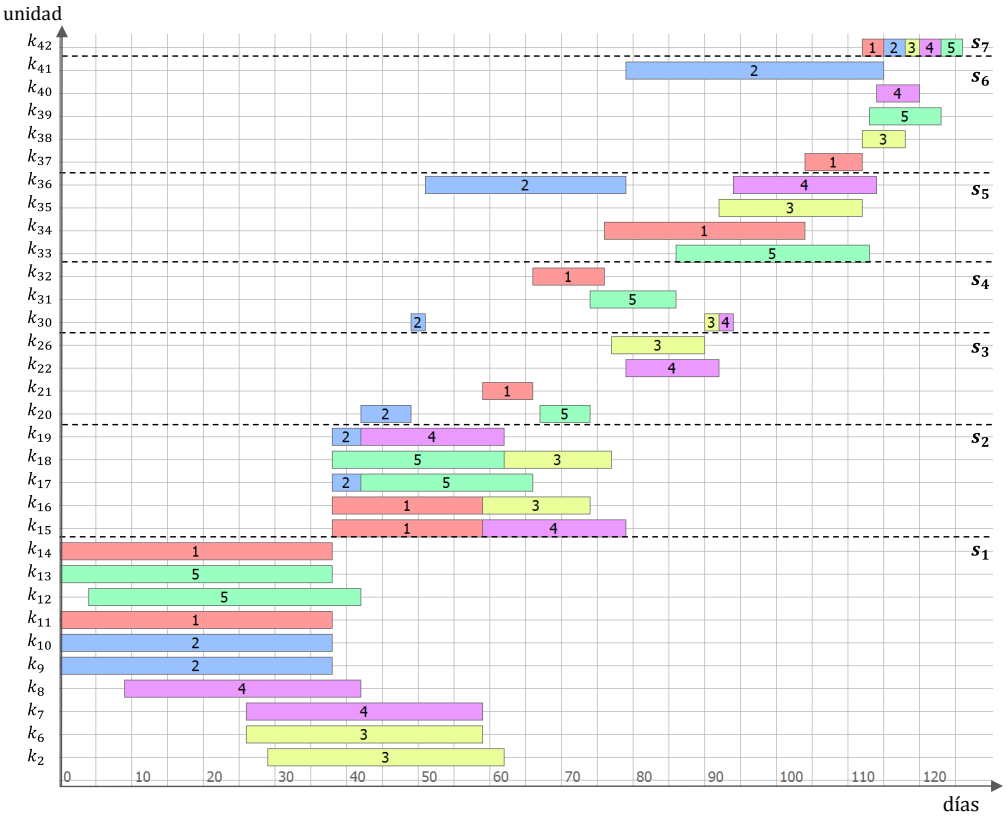



Figura 3.7. Ejemplo 3.1 – Diagrama Gantt del "schedule" óptimo reportado por la estrategia de solución para P.02 - $MK = 126{,}3$ días.

Por otra parte, es importante resaltar que para instancias de tamaño mediano a grande, el enfoque exacto no converge a la solución óptima dentro del límite de tiempo predefinido (3600 CPUs), mientras que el procedimiento iterativo obtiene soluciones de buena calidad en pocos segundos de tiempo de CPU para todas las instancias consideradas. Aunque el enfoque iterativo no puede asegurar la optimalidad de las soluciones, es capaz de obtener mejores soluciones para problemas de programación de tamaño mediano a grande (P.03-P.10), en hasta un 15,2%, que las reportadas por el enfoque matemático riguroso y con un muy bajo esfuerzo computacional. Por ejemplo, para casos de tamaño mediano el algoritmo puede mejorar la solución obteniendo una reducción importante de makespan, mayor al 3,6%, con un esfuerzo computacional de aproximadamente 40 CPUs. Para un tamaño mayor del problema, como la instancia P.09, el método exacto logra alcanzar un makespan final de 290,8 días con un valor de GAP relativo del 37,9% después de 1 hora de tiempo de CPU, mientras que el algoritmo reporta una solución igual a 262,8 días en sólo 49,8 segundos de CPU, lo que se refleja en una mejora del valor de la solución del 9,6%. Es importante destacar que, esta mejora en el plan de operaciones representa un mes de trabajo en el sistema productivo, y por lo tanto, un ahorro significativo para el mismo, tanto en costos, como en alguna posible penalidad.

La gran diferencia de tiempo computacional presentada por el enfoque de descomposición en las distintas instancias, se debe a la capacidad del método iterativo de descomponer el problema en sub-problemas de menor tamaño y ofrecer, de esta manera, una mejora sustancial en el rendimiento computacional para resolver problemas de escala industrial. Esta situación se puede observar fácilmente en los ejemplos donde la cantidad de bloques es mayor a 5, ya que el enfoque matemático consume considerablemente más tiempo de CPU que la estrategia de descomposición. Por lo tanto, se puede concluir que para casos de tamaño reducido, el modelo MILP puede alcanzar la solución óptima, pero cuando se trata de problema con una complejidad moderada a alta, se encuentran soluciones deficientes en un período de tiempo corto.

Para ilustrar la complejidad del proceso de ensamblado de bloques, los mejores programas de operaciones reportados por el algoritmo iterativo, para instancias de tamaño mediano y grande, se muestran de manera gráfica en las Figuras 3.8 y 3.9, respectivamente. En el primer caso, se puede observar (Figura 3.8) el diagrama Gantt correspondiente a la mejor solución encontrada para la instancia P.05 (15 bloques, 30 sub-bloques y política de almacenamiento UIS) de tamaño mediano. Mientras que la Figura 3.9 muestra el mejor programa de operaciones reportado para un ejemplo de tamaño industrial, como P.10, que involucra 25 bloques y 50 sub-bloques, y adopta una política de almacenamiento intermedio tipo NIS.

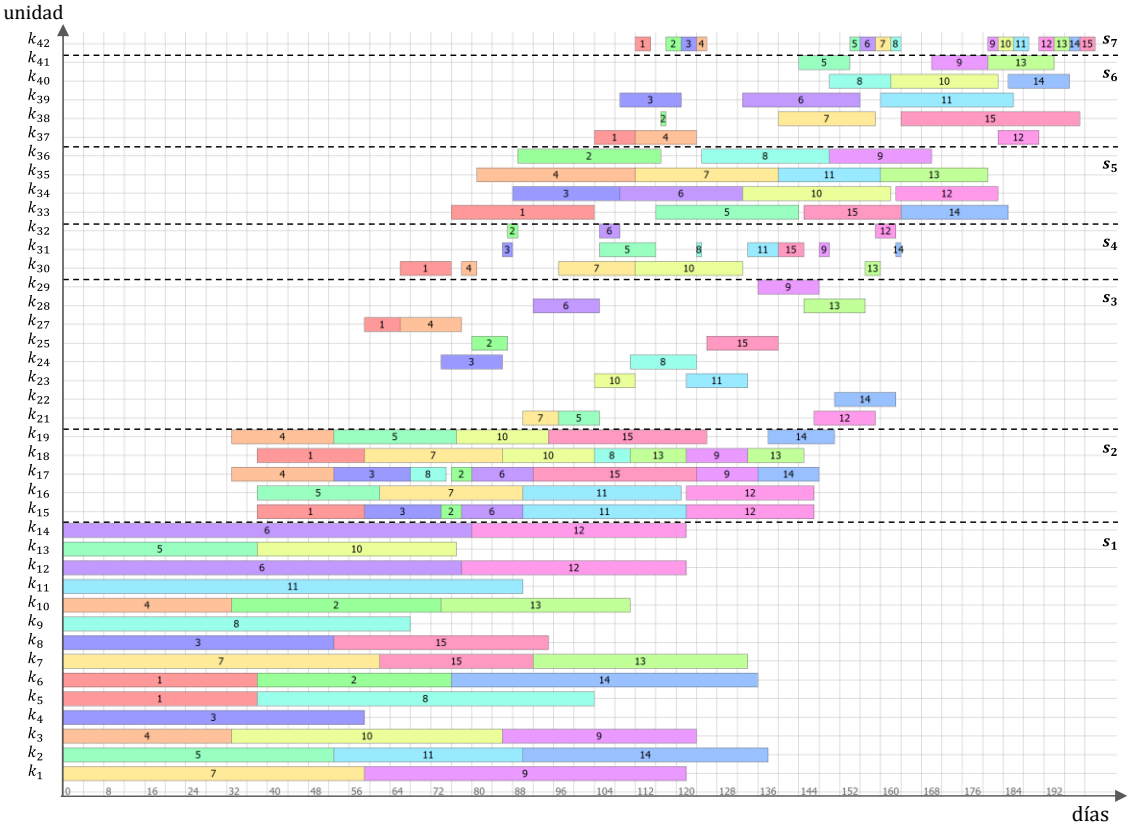



Figura 3.8. Ejemplo 3.1 – Diagrama Gantt del mejor "schedule" reportado por la estrategia de solución para P.05 - $MK = 200{,}2$ días.

Teniendo en cuenta el caso P.10, se puede apreciar cómo el makespan disminuye considerablemente de 319,1 a 270,5 días, con una reducción del 15,2%, con respecto al obtenido a partir del enfoque riguroso y tiempo límite de 1 hora (graficado en la Figura 2.7

del Capítulo 2). Mientras que se alcanza una reducción del 5% (pasando de 284,7 a 270,5 días) al considerar la solución reportada por dicho modelo MILP para el mismo problema $N \times M$, pero sin límite de tiempo computacional (Figura 2.8).

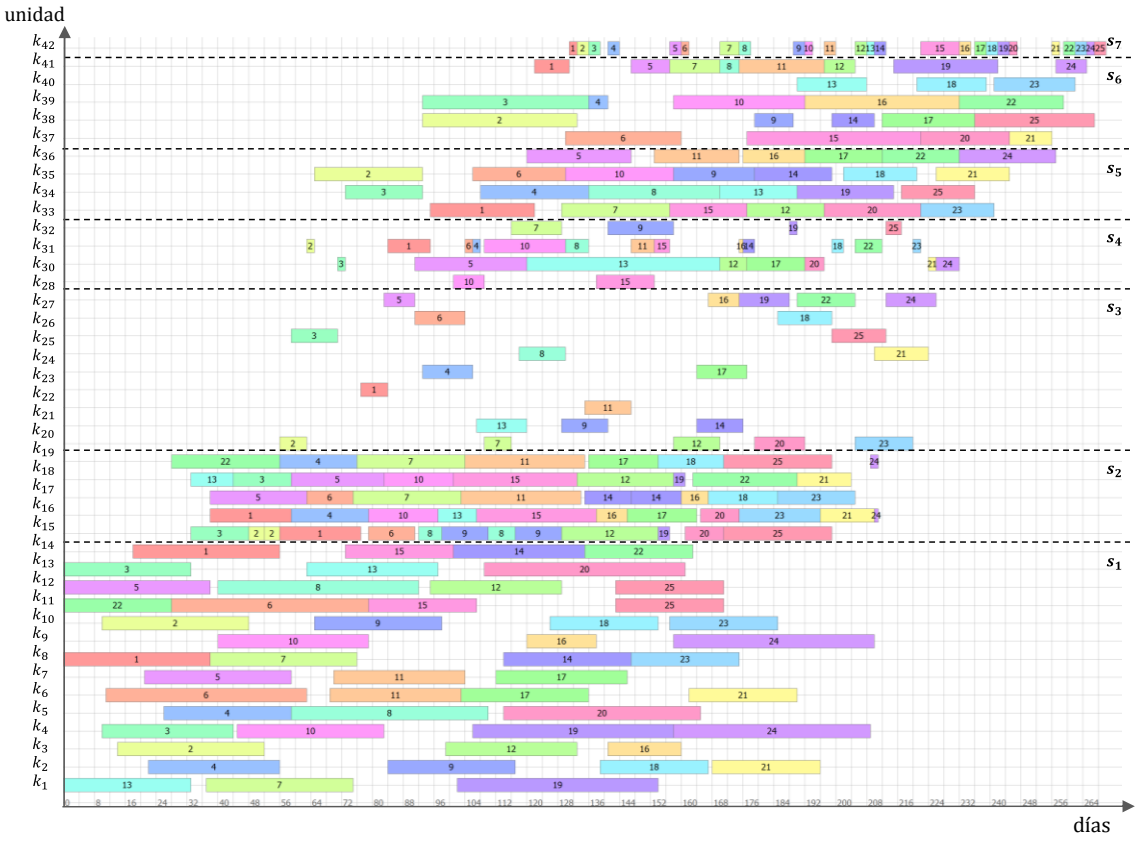



Figura 3.9. Ejemplo 3.1 – Diagrama Gantt del mejor "schedule" reportado por la estrategia de solución para P.10 - $MK = 270{,}5$ días.

Notar que en todos los diagramas, el par de sub-bloques que conforman un determinado bloque, se identifican con el número y color de dicho bloque. Además, se puede apreciar que, en los tres programas graficados en esta sección, no se utilizan todos los recursos disponibles del astillero (unidades correspondientes de cada etapa de procesamiento), especialmente en las instancias que involucran una menor cantidad de bloques (Figura 3.7 y 3.8).

### 3.4.2. Ejemplo 3.2

La estrategia de solución propuesta se utiliza una vez más como herramienta para resolver la programación de las operaciones de producción y ensamblado de un caso de estudio similar al analizado en el Ejemplo 3.1. Para este ejemplo se considera una nueva configuración del sistema productivo correspondiente a otro astillero. Tal configuración difiere en la cantidad de unidades en paralelo en las etapas de procesamiento y además, considera una etapa inicial adicional. Asimismo, se considera el montaje de una embarcación de tamaño superior, lo que implica un diseño modular que involucra una mayor cantidad de bloques y sub-bloques. Desde el punto de vista de la eficiencia que debe reflejar una herramienta para abordar problemas de "scheduling" FFSP a nivel industrial, este ejemplo intenta mostrar nuevamente, desde lo numérico, la superioridad computacional y flexibilidad del enfoque de descomposición propuesto.

De esta manera, se parte de un sistema productivo FFSP que consiste en 8 etapas de procesamiento con un total 21 unidades idénticas en paralelo, para elaborar un barco a partir del ensamblado de 33 grandes bloques, cada uno de los cuales está conformado por 2 sub-bloques de características similares, tal y como se consideró en el primer ejemplo. La estructura general del proceso productivo se muestra en la Figura 3.10. Considerando esta nueva configuración, se debe determinar el mejor "schedule" que minimice el makespan, según la secuencia de montaje final de la embarcación bajo estudio, representada por el parámetro $Seq_i$.

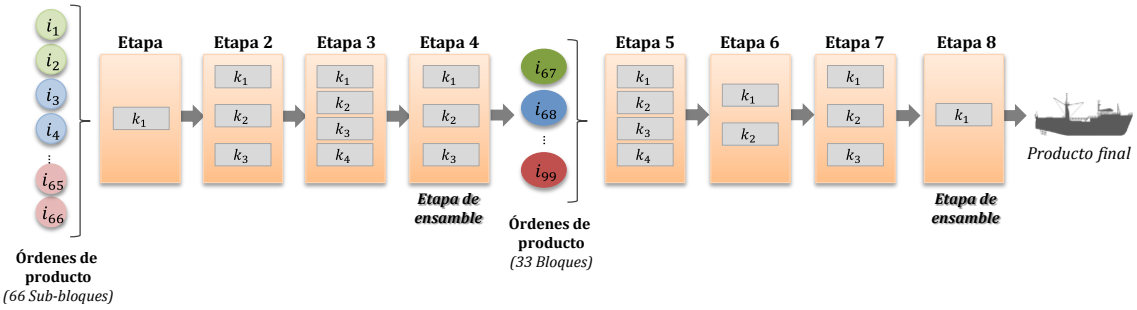



Figura 3.10. Estructura del proceso productivo – Ejemplo 3.2.

Al igual que para el Ejemplo 3.1, se definen y analizan diferentes instancias del caso de estudio, donde varía el tamaño del problema, es decir, se consideran diferentes cantidades de ensambles y subensambles ($N \times M$). Estas son resueltas, tanto por el enfoque de optimización basado en el concepto de precedencia general global, como por el algoritmo iterativo presentado en este capítulo. La Tabla 3.3 presenta las diferentes instancias consideradas, los correspondientes requerimientos computacionales y las soluciones obtenidas a partir de ambos enfoques, dentro del límite de tiempo predefinido de 1 hora de CPU. La última columna de dicha tabla, relativa al porcentaje de mejora que ofrece la estrategia de solución propuesta, muestra la eficiencia computacional y superioridad de dicho enfoque frente a las formulaciones matemáticas puras de optimización. Además, los tamaños de los modelos MILP para cada una de las instancias se resumen en la Tabla 3.4.

Tabla 3.3. Comparación entre el modelo matemático y el método iterativo – Ejemplo 3.2.

| Instancias | Tamaño $N \times M$ | Formulación basada en precedencia general | | | Estrategia de solución basada en MILP | | | Mejora en la solución (%) |
|---|---|---|---|---|---|---|---|---|
| | | Solución MILP | GAP | CPU time (s) | Solución inicial | Mejor solución | Tiempo total CPU (s) | |
| P.11 | 18 × 36 | 377 | 37,0 | 3600 | 366 | 353 | 7,6 | 6,4 |
| P.12 | 24 × 48 | 541 | 56,7 | 3600 | 473 | 421 | 12,8 | 22,2 |
| P.13 | 33 × 66 | 728 | 62,0 | 3600 | 551 | 516 | 83,2 | 29,1 |

Tabla 3.4. Tamaño de los modelos MILP basados en el enfoque de precedencia general – Ejemplo 3.2.

| Instancias | Variables binarias | Variables continuas | Restricciones |
|---|---|---|---|
| P.11 | 7551 | 434 | 14995 |
| P.12 | 13308 | 578 | 26617 |
| P.13 | 24981 | 1157 | 50267 |

A partir de la Tabla 3.4, en este ejemplo también se puede observar que, el número de variables binarias de secuenciación y asignación aumenta considerablemente al aumentar el número de órdenes de pedido, es decir, al requerir el procesamiento de una mayor cantidad de bloques y sub-bloques. El método exacto reporta soluciones factibles pero con un gran valor de GAP, llegando a un valor del 62% en el caso de mayor tamaño (P.13). La formulación MILP del modelo para dicha instancia, la cual corresponde al caso de estudio real (33 ensambles × 66 subensambles), involucra 50267 restricciones, 24981 variables binarias y 1157 variables continuas. La solución reportada en 1 hora de tiempo de CPU corresponde a un makespan de 728 días, frente a la solución encontrada por la metodología iterativa de 516 días para el mismo caso, significando un 29,1% de mejora, en menos de 2 minutos de tiempo computacional. Las respectivas soluciones se ilustran a través de diagramas Gantt en las Figuras 3.11 y 3.12, respectivamente. Dichos diagramas ilustran la secuencia de producción sobre las distintas unidades de procesamiento de cada etapa. Debido a que se considera un "scheduling" detallado del procesamiento de las órdenes de producto, la secuencia de producción de éstas puede no seguir el mismo orden en todas las etapas.

De manera similar a como se efectúo en el ejemplo anterior, también se intenta resolver el problema utilizando el modelo MILP monolítico sin establecer un límite de tiempo, es decir, sólo considerando un criterio de terminación del 0% de valor de GAP. Sin embargo, en todos los casos el solucionador termina al exceder la capacidad de memoria. Por ejemplo, para la instancia P.13, correspondiente al caso de estudio real, a pesar de obtener un makespan de 600 días con una brecha de optimalidad del 55%, el enfoque exacto no se logra mejorar la solución con respecto a la reportada por el algoritmo iterativo (516 días). En otras palabras, la estrategia de solución sigue superando a la formulación matemática pura ampliamente, tanto en tiempo de resolución, como en la calidad de solución obtenida (en un 14%).

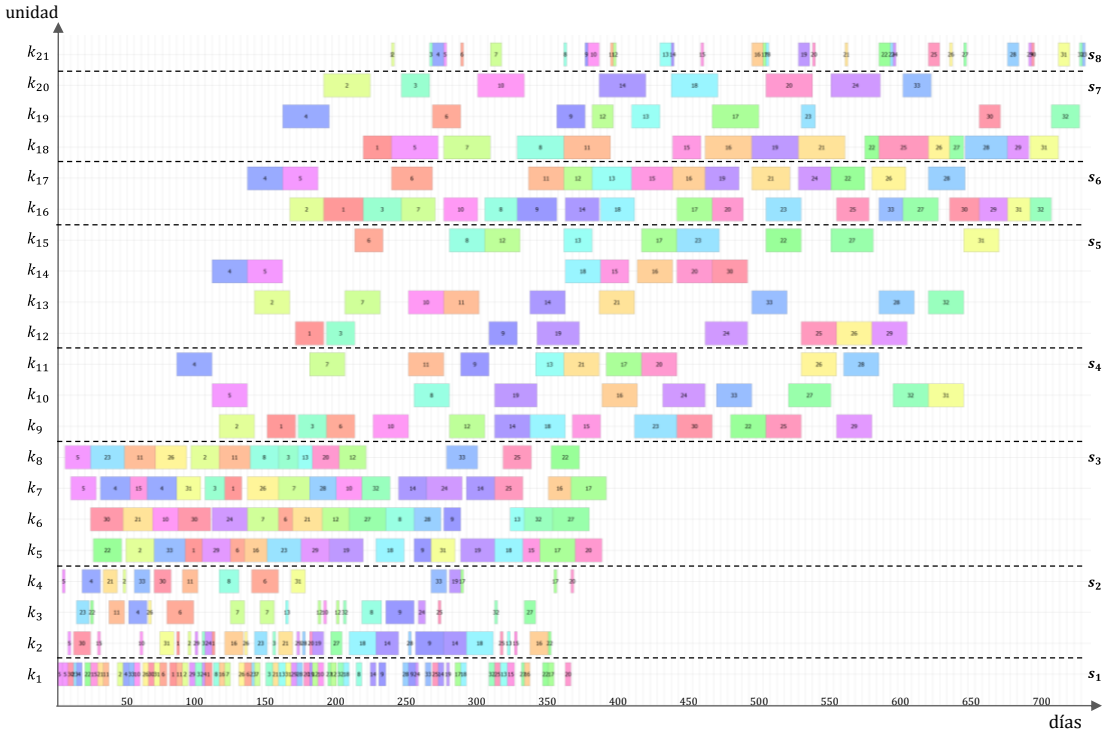



Figura 3.11. Ejemplo 3.2 - Diagrama Gantt del "schedule" reportado por el modelo MILP para P.13 - $MK = 728$ días.

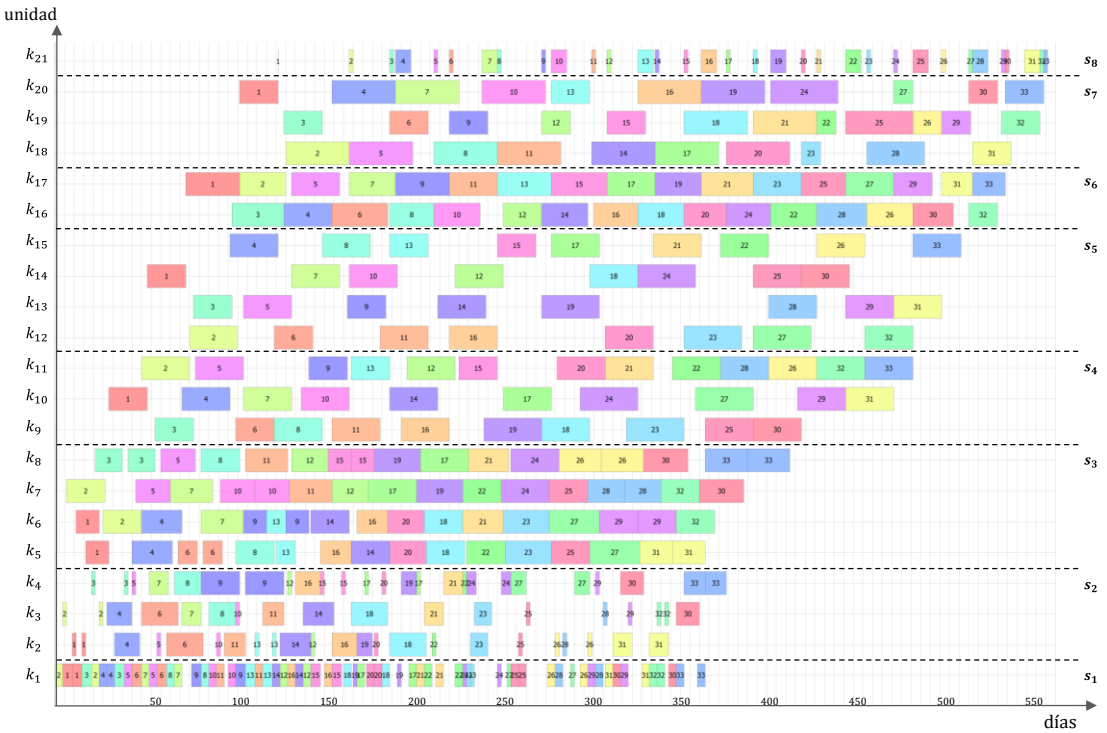



Figura 3.12. Ejemplo 3.2 - Diagrama Gantt del "schedule" reportado por la estrategia de solución para P.13 - $MK = 516$ días.

Los resultados alcanzados durante la experimentación demuestran que:

i. En comparación con los resultados obtenidos en el capítulo anterior, se puede deducir que al tratarse de problemas NP-hard, el esfuerzo computacional de los enfoques de optimización exactos crece exponencialmente con el tamaño del problema, resultando ser inmanejables para casos de estudio de tamaño industrial debido al gran número de decisiones binarias y restricciones involucradas.

ii. El algoritmo iterativo propuesto en este capítulo mostró una mayor eficiencia, con respecto a los métodos exactos, para problemas de gran tamaño y complejidad, reportando una solución factible y de alta calidad en tan sólo unos pocos segundos de tiempo de CPU, en todos los casos evaluados. En otras palabras, para problemas de escala industrial, resulta muy conveniente descomponer el problema de "scheduling" y resolver iterativamente modelos MILP de tamaño reducido asociados, en lugar de recurrir a un modelo MILP monolítico.

## 3.5. Conclusiones

En este capítulo se desarrolló una estrategia de descomposición basada en una formulación MILP para dar solución a problemas de programación de operaciones FFSP de tamaño real, que involucran operaciones de ensamblado. En el anterior capítulo de esta tesis se pudo demostrar que para la resolución de este tipo de problemas, la eficiencia computacional que presentan los enfoques de espacio completo, como las formulaciones matemáticas MILP, se deteriora rápidamente a medida que aumenta el tamaño del mismo. Con el objetivo de salvar dicha limitación, se utilizó como base el modelo MILP de precedencia general presentado en tal capítulo, para proponer un procedimiento iterativo de dos pasos: (i) la primera fase tiene como fin la obtención de una solución inicial factible con un mínimo requerimiento computacional (*etapa constructiva*); (ii) la segunda fase mejora gradualmente dicha solución aplicando iteraciones de "rescheduling" (*etapa de mejora*). De esta manera, el algoritmo consiste en una estrategia de descomposición

sistemática, que busca resolver en cada iteración versiones reducidas (en tamaño) del modelo MILP, intentando mantener el número de variables del problema en un nivel razonable. En otras palabras, se propone combinar las principales ventajas de los modelos de optimización, como la rigurosidad, con la flexibilidad de los procedimientos heurísticos, tales como técnicas de descomposición-agregación y procedimientos de mejora, haciendo posible generar y mejorar progresivamente la solución obtenida para problemas de gran tamaño en un tiempo computacional aceptable.

Para la validación de la metodología propuesta, se resolvió un conjunto de pequeñas instancias derivadas del caso de estudio real de una compañía naviera dedicada a la construcción de barcos a partir de procesos de ensamblado de bloques, planteado en la sección 2.4.1 del Capítulo 2. Asimismo, se evaluó el desempeño del procedimiento iterativo para instancias de mayor tamaño del mismo caso de estudio. Por otro lado, se analizó un nuevo caso de estudio correspondiente a la construcción de una embarcación, que involucra una mayor cantidad de ensambles y subensambles, la cual se lleva a cabo en un astillero con una configuración diferente a la del anterior ejemplo resuelto. En todos los casos se minimiza el tiempo total requerido para la obtención del producto final.

A partir de los ejemplos analizados se puso de manifiesto que la estrategia de solución propuesta permite reducir, de manera significativa, los requerimientos computacionales para resolver problemas de "scheduling" detallado. Esto se vio reflejado en las mejoras observadas en las soluciones reportadas, entre los métodos exactos y el algoritmo iterativo, alcanzando reducciones considerables en la función objetivo, de hasta un 29% para las instancias de mayor tamaño, con un mínimo esfuerzo computacional, equivalente a pocos segundos de tiempo de CPU.

De esta manera, se puede concluir que, aunque la metodología de descomposición propuesta en este capítulo no permite garantizar la optimalidad de las soluciones reportadas, demuestra ser robusta, flexible y altamente eficiente, capaz de adaptarse y resolver problemas industriales de tamaño real.

# *Nomenclatura*

## *Conjuntos*

| | |
|---|---|
| $I$ | conjunto de órdenes de productos (índice $i, i'$) |
| $I^b$ | conjunto de productos ensamblados (o bloques) |
| $I^{sb}$ | conjunto de partes, subensambles o sub-bloques |
| $SB_i$ | conjunto de partes o subensambles que conforman un producto ensamblado $i \in I^b$ |
| $S$ | conjunto de etapas de procesamiento (índice $s, s'$) |
| $S_i$ | conjunto de etapas que procesan las órdenes de producto $i \in I$ |
| $S^a$ | conjunto de etapas de procesamiento $s$ disponibles para procesar los subensambles $i \in SB_i$ |
| $S^a_{seq}$ | conjunto de etapas de ensamblado que siguen una secuencia de montaje predefinida |
| $K$ | conjunto de unidades de procesamiento |
| $K_s$ | conjunto de unidades en paralelo de la etapa de procesamiento $s$ |

## *Parámetros*

| | |
|---|---|
| $tp_{is}$ | tiempo de procesamiento de la orden de producto $i$ en la etapa $s$ |
| $Seq_i$ | orden en que se deben ensamblar los productos $i$ |
| $M$ | constante big-M |
| $iter$ | número de orden que se inserta en cada iteración |

$active_i$ orden de producto $i$ activa en cada iteración algorítmica

$sY_{ik}$ parámetro para las decisiones de asignación del producto $i$ en la unidad $k$

$sW_{ii'k}$ parámetro para las decisiones de secuenciación de los productos $i$ e $i'$en la unidad $k$

$Sol_{inicial}$ solución inicial factible reportada por la etapa de construcción del algoritmo de descomposición

$BestSol$ mejor solución encontrada en la etapa de mejora del algoritmo de descomposición

$CurrentSol$ solución encontrada en cada iteración de la etapa de mejora

***Variables Continuas***

$Ts_{is}$ tiempo de inicio de procesamiento del producto $i$ en la etapa $s$

$Tf_{is}$ tiempo de finalización del procesamiento del producto $i$ en la etapa $s$

$MK$ makespan

***Variables binarias***

$W_{ii'k}$ denota si la orden de producto $i$ se procesa antes que la orden $i'$ en la unidad de procesamiento $k$

$Y_{ik}$ determina si la orden de producto $i$ se procesa en la unidad de procesamiento $k$

# Capítulo 4

# *Metodología integrada de programación detallada de las operaciones y rediseño de plantas de manufactura flexible*

## 4.1. Introducción

Como se ha detallado en el primer capítulo de esta tesis, encontrar la mejor programación de operaciones depende principalmente del entorno productivo, las restricciones del proceso y el indicador de rendimiento establecido (Pinedo, 2016). Uno de los problemas más difíciles en esta área es el de programación de sistemas de manufactura job shop (JSP) y su variante job shop flexible (FJSP), donde se utiliza una cantidad limitada de recursos para procesar un conjunto de tareas o productos especializados, cada uno con su secuencia propia de procesamiento. Estos problemas tienen como objetivo encontrar la secuencia apropiada de operaciones en cada máquina, de manera tal de optimizar el indicador de rendimiento. Un indicador clásico en los problemas JSP es el makespan, es decir, el tiempo necesario para completar todas las tareas o fabricar todos los productos requeridos. Cabe recordar que, el FJSP es una

extensión del JSP clásico, donde las operaciones pueden procesarse en un conjunto de máquinas disponibles.

En particular, los problemas de programación FJSP surgen en el área de sistemas de manufactura flexibles. De esta manera, debido a la variedad de situaciones reales que representan y por lo tanto, a su amplia aplicación en entornos industriales, tanto los JSP como los FJSP, han recibido una importante atención de la comunidad científica. Ambos son problemas de optimización con una alta complejidad combinatoria, considerados problemas NP-hard (Garey et al., 1976), por lo que hasta el momento no hay enfoques determinísticos para resolver de forma óptima instancias de problemas de tamaño real. Por este motivo, los problemas JSP y FJSP han sido objeto de un estudio exhaustivo y en las últimas décadas, se han propuesto numerosas técnicas de distinta naturaleza para su resolución. En particular, el esfuerzo se suele centrar en metaheurísticas y métodos heurísticos que, a pesar de no garantizar la optimalidad de la solución reportada, suelen proporcionar soluciones eficientes, es decir muy buenos “schedules", en un tiempo de cómputo reducido.

Particularmente, la alta complejidad combinatoria de este tipo de problemas se debe a que se deben enfrentar dos sub-problemas de manera simultánea: (i) la asignación de tareas y (ii) la secuenciación de las mismas, respectivamente. El problema FSJP es definido por primera vez por Brucker y Schlie (1990). Sin embargo, Brandimarte y Torino (1993) fueron los primeros en abordar este problema teniendo en cuenta ambos aspectos (decisiones de asignación y secuenciación) a través de un enfoque jerárquico, donde se resuelve primero el sub-problema de asignación, a partir del uso de algunas reglas de despacho, y luego, el sub-problema de secuenciación, utilizando una heurística de “tabu-search”. Hurink et al. (1994) también utilizaron una heurística de “tabu-search” para considerar la reasignación y el reordenamiento de manera independiente. Posteriormente, Dauzère-Pérès y Paulli (1997) introdujeron enfoques integrados, donde consideraron ambas decisiones de manera simultánea.

En los siguientes años, varias investigaciones relevantes se centraron en resolver el problema de programación FJSP. Por ejemplo, Xia y Wu (2005) propusieron un enfoque de solución jerárquica para resolver un problema FJSP multiobjetivo. Luego, Saidi-Mehrabad y Fattahi (2007) presentaron un modelo matemático y un algoritmo con dos heurísticas para resolver el problema FJSP con "set-ups" dependientes de la secuencia, donde el primero asigna cada operación a una máquina de un conjunto de máquinas y el segundo secuencia las operaciones asignadas en todas las máquinas con el fin de obtener un "schedule" factible que minimice el makespan. Asimismo, Fattahi et al. (2009, 2007) mostraron que los algoritmos jerárquicos tienen un mejor rendimiento sobre los integrados.

Por otra parte, Blazewicz et al. (1991) fueron los primeros en enfocarse en modelos matemáticos para problemas de programación, presentando formulaciones matemáticas para dichos problemas, con máquinas simples, máquinas paralelas y entornos job shop. Luego, Pan (1997) comparó diferentes modelos matemáticos para los problemas de "scheduling" tipo job shop y flow shop. Recientemente, Lee y Maravelias (2017a) presentaron dos formulaciones matemáticas MILP para el procesamiento simultáneo de lotes de productos y la programación de operaciones en plantas multipropósito con limitaciones de almacenamiento. Otros autores, como Demir y Kürşat Işleyen (2013) y Roshanaei et al. (2013), también propusieron modelos de programación matemática lineal mixta-entera para FJSP. Sin embargo, como se ha mencionado anteriormente, debido a la complejidad del problema, se han investigado principalmente metaheurísticas.

Además de la programación de operaciones, existe el problema de diseño que, conceptualmente involucra la determinación, tanto de la dimensión de tanques de almacenamiento y unidades de procesamiento, como de la estructura de red del proceso. En la literatura, existen trabajos que han considerado el problema de "scheduling" y de diseño de manera integrada para plantas multietapa multiproducto (Fumero et al., 2012; Voudouris y Grossmann, 1993). En particular, para plantas multipropósito las

publicaciones se focalizaron en cada caso particular debido a la alta complejidad que presentan (Castro et al., 2005; Chibeles-Martins et al., 2010; Corsano et al., 2007; Lin y Floudas, 2001). Sin embargo, estas propuestas no permiten evaluar posibles reconfiguraciones de los equipos o unidades de procesamiento en plantas ya existentes, que se encuentran en funcionamiento con una configuración predefinida. Teniendo en cuenta estas limitaciones, y con el propósito de afrontar el desafío de integrar las dos problemáticas, el presente capítulo aborda el problema integrado de "scheduling" y rediseño de sistemas de manufactura flexible, donde existen unidades multipropósito y se requieren operaciones de ensamblado en diferentes etapas de proceso.

En primer lugar, se presenta un modelo MILP basado en el concepto de precedencia general. Luego, dicha formulación matemática se utiliza como base para el desarrollo de un algoritmo de descomposición, con el fin de abordar de manera eficiente los problemas de gran escala que surgen en plantas de manufactura flexible. El principal objetivo es desarrollar una metodología para determinar el mejor "schedule" y analizar la configuración de la planta, con el objetivo de minimizar el tiempo máximo de procesamiento (makespan) y maximizar la eficiencia del uso del equipamiento en un entorno job shop flexible. Desde el punto de vista del rediseño de la planta, se evalúa: (i) la existencia de estaciones de trabajo sobredimensionadas, (ii) posibles reubicaciones factibles de equipos, (iii) las operaciones debe realizar cada equipo, correspondientes a una etapa de procesamiento determinada o a más de una etapa.

La estrategia de resolución desarrollada permite:

- Determinar una buena solución inicial al problema de "scheduling", en un tiempo razonable, incorporando criterios de inserción, asignación y pre-ordenamiento en la formulación matemática original del problema. Dichos criterios están basados en el algoritmo presentado en el Capítulo 3.

- Mejorar la solución inicial iterativamente mediante la ejecución de un conjunto limitado de cambios permitidos en la secuenciación de tareas y en la asignación

de los recursos de procesamiento. El número de cambios simultáneos que se pueden llevar a cabo se incrementa gradualmente, en función del tiempo computacional requerido. Las principales modificaciones que se permiten en cada iteración están vinculadas a: (i) cambios en los tiempos iniciales y finales de las tareas de procesamiento, (ii) un conjunto de cambios simultáneos en la asignación de tareas a unidades y (iii) un conjunto de cambios simultáneos permitidos en la secuenciación de las tareas asignadas a un dado recurso.

- Detener el procedimiento iterativo cuando no se observa mejora en el programa de operaciones.

- Evaluar posibles sobredimensionamientos de celdas de trabajo, fijando el valor de la solución obtenida ($MK$) y reduciendo las unidades de procesamiento requeridas para cumplir dicha solución.

- Evaluar posibles reubicaciones de las unidades que presentan menos utilización, tanto para propósito único, como para múltiples propósitos, según los cambios factibles que se puedan considerar en la planta bajo estudio, y de esta manera, mejorar el uso de los recursos y el valor del makespan.

El capítulo está organizado de la siguiente manera. En la sección 4.2, se presenta la definición genérica del problema. La sección 4.3 presenta el nuevo modelo matemático MILP desarrollado para resolver el problema de "scheduling" de plantas job shop flexibles. En la sección 4.4, se describe el algoritmo iterativo basado en el modelo de optimización mencionado anteriormente, y que tiene como objetivo resolver de manera integrada el problema de programación de las operaciones y de rediseño. La factibilidad de dicho algoritmo se demuestra con un ejemplo ilustrativo en la sección 4.5. Asimismo, en dicha sección se extiende la aplicabilidad de la estrategia de solución propuesta resolviendo dos estudios de casos, uno basado en la literatura y otro en un caso real. Por último, las conclusiones finales se detallan en la sección 4.6 y seguidamente, se presenta la nomenclatura utilizada.

## 4.2. Descripción del problema

Para el problema abordado en este capítulo se considera una planta multietapa multiproducto compuesta por $S$ etapas de procesamiento, de las cuales algunas se encuentran destinadas a efectuar operaciones de ensamblado. Al igual que en el Capítulo 3, estas etapas se pueden discriminar según el tipo de montaje que llevan a cabo y de las restricciones operativas que se deben cumplir para dicha operación: (i) $s \in S^a$ y (ii) $s \in S^a_{seq}$, donde $S^a \subset S$ y $S^a_{seq} \subset S$ (ver sección 2.2.2). Asimismo, cada etapa cuenta con unidades de procesamiento en paralelo $k \in K_s$, las cuales pueden ser de propósito único o multipropósito, es decir, pueden llevar a cabo las operaciones correspondientes a una única etapa o a más de una. Dichas unidades procesan un conjunto de productos $i \in I$, de los cuales algunos de ellos se consideran productos intermedios o subensambles ($i \in I^a$), que posteriormente forman ensambles. Notar que, dependiendo de la cantidad de etapas de ensamblado que conforman el sistema productivo, estos ensambles pueden luego, ser considerados productos finales $i \in I^f$ o tener que pasar por operaciones de montaje para ser unidos y obtener el resultado final. En otras palabras, puede existir más de una etapa de ensamblado en una receta de procesamiento. Además, se debe considerar que cada producto $i$ tiene una configuración propia, es decir, una secuencia de procesamiento y por lo tanto, un proceso de fabricación diferente.

A modo de ejemplo ilustrativo, la Figura 4.1 muestra un problema de programación de operaciones de producción y ensamblado con unidades multipropósito, en el que se debe procesar un conjunto de 9 órdenes de producto. Notar que, aquellas unidades que realizan funciones similares son agrupadas en una misma celda o estación de trabajo (también llamada "workstation"), denotadas por $u \in U$. Las dos primeras celdas de trabajo, $u_1$ y $u_2$, procesan los productos desde $i_1$ a $i_7$, donde 6 de ellos son ensamblado en $u_2$ para generar los productos ensamblados $i_8$, $i_9$ y $i_{10}$. Luego, estos tres ensambles son procesados en las estaciones $u_4$ y $u_5$, para obtener los productos finales. En cambio, el producto $i_6$ no requiere operaciones de ensamblado en su receta de procesamiento.

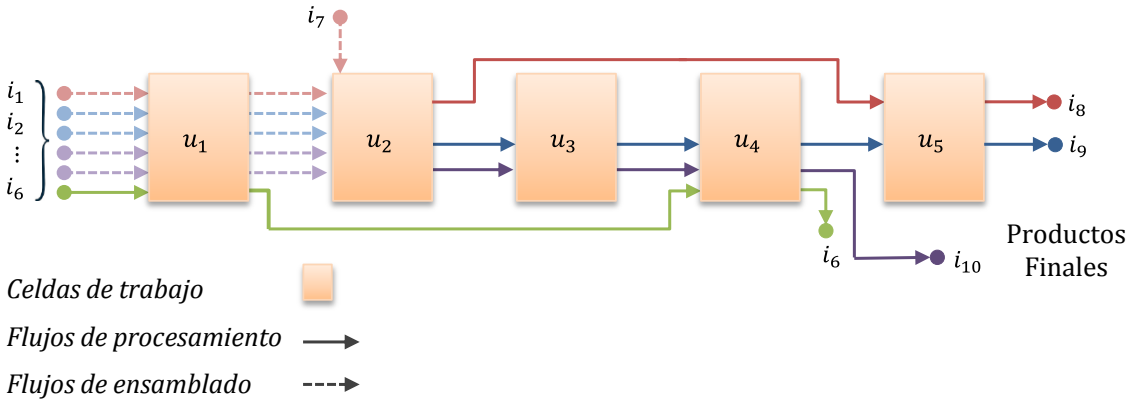



Figura 4.1. Proceso productivo con unidades multipropósito y operaciones de ensamblado.

El objetivo principal del problema es encontrar un programa de operaciones que optimice la función objetivo establecida, en este caso definida como el makespan, dada la secuencia de unidades o receta de procesamiento de cada producto. De este modo, para el problema de "scheduling" con decisiones de rediseño de plantas de manufactura flexible, donde se tienen en cuenta unidades multipropósito y diferentes operaciones de ensamblado, se considera el modelado de un sistema *job shop flexible* con las siguientes características:

- Existe un conjunto de productos $i \in I$ que deben procesarse siguiendo una secuencia predefinida de etapas de procesamiento, o también llamada receta de procesamiento, donde $s \in S_i$. Por lo tanto, no necesariamente se utilizan todas las etapas para el procesamiento de un producto.

- Se define un conjunto de estaciones de trabajo o "workstations" $u \in U$, cada una de las cuales está compuesta por un grupo de unidades de procesamiento $k \in K_u$, que tienen la particularidad de realizar operaciones similares o idénticas, correspondientes a una o más etapas productivas.

- Se considera que las unidades de procesamiento operan sin fallas y no necesitan mantenimiento preventivo.

- Existe un conjunto de equipos o unidades de procesamiento $K = \{k_1, k_2, .., |K|\}$.

- Cada unidad o equipo $k \in K$ puede procesar un único producto a la vez.

- Se modela un subconjunto de etapas de procesamiento $s \in S_u$ que se pueden llevar a cabo en la estación $u \in U$. Asimismo, también se define un subconjunto de unidades $K_s \subset K$ destinadas a realizar las operaciones de cada etapa de procesamiento $s \in S$.

- Análogamente a los capítulos anteriores, los productos finales $i \in I^f$ pueden ser el resultado del ensamblado de productos intermedios $i' \in I^a$. Por lo tanto, se define un subconjunto $SA_i$ que contiene todos los subensambles $i'$ del producto $i$, donde $I = (I^a \cup I^f)$.

- Existen etapas de procesamiento $s \in S^a$ y $s \in S^a_{seq}$ que realizan diferentes tipos de operaciones de ensamblado de productos (ver sección 2.2.2).

- El tiempo de procesamiento del producto $i$ en la etapa $s$, $tp_{is}$, es conocido en todos los casos.

- Los tiempos de transporte entre etapas se consideran insignificantes o se incluyen en los tiempos de procesamiento, por lo tanto, no se tiene en cuenta de manera independiente.

- Todos los parámetros del modelo son determinísticos.

- Se puede adoptar una política de almacenamiento intermedio entre etapas o "workstations" tipo UIS o NIS.

Debido a que, como se mencionó anteriormente, las unidades o equipos se pueden clasificar según sean de propósito único o multipropósito, se exponen las siguientes consideraciones a tener en cuenta en el modelado del problema:

- Se define un conjunto de tareas $T = \{t_1, t_2, \dots, |T|\}$ que contiene todas las operaciones correspondientes a la receta de cada producto a procesarse.

- Por lo tanto, cada producto $i \in I$ se descompone en un conjunto de tareas $t \in JT_i$, donde $JT_i \subset T$.

- Análogamente, se establece un subconjunto $ST_s \subset T$ para incluir todas las tareas que se realizan en cada etapa de procesamiento $s \in S$.

- Para cada producto $i$ procesado en la etapa $s \in S_i$, existe una tarea $t$, tal que $t \in (JT_i \cap ST_s)$.

- Por consiguiente, cada tarea $t \in T$ queda definida por un producto y una etapa de procesamiento $(i, s)$.

A continuación, en la Figura 4.2, se ilustra el programa de operaciones de un pequeño ejemplo de 4 productos ($i_1$-$i_4$) y 2 etapas de procesamiento ($s_1, s_2$), a fin de visualizar la definición de los subconjuntos $JT_i$ y $ST_s$. Notar que, por ejemplo, para el procesamiento del producto $i_1$, que debe pasar por ambas etapas, se definen dos tareas de procesamiento asociadas a cada etapa: $t_1$ a la etapa $s_1$ ($t_1 \in ST_{s_1}$) y $t_5$ a la etapa $s_2$ ($t_5 \in ST_{s_2}$). Asimismo, ambas tareas forman parte del conjunto de operaciones en que se descompone el procesamiento del producto $i_1$ en el sistema: $(t_1, t_5) \in JT_{i_1}$.

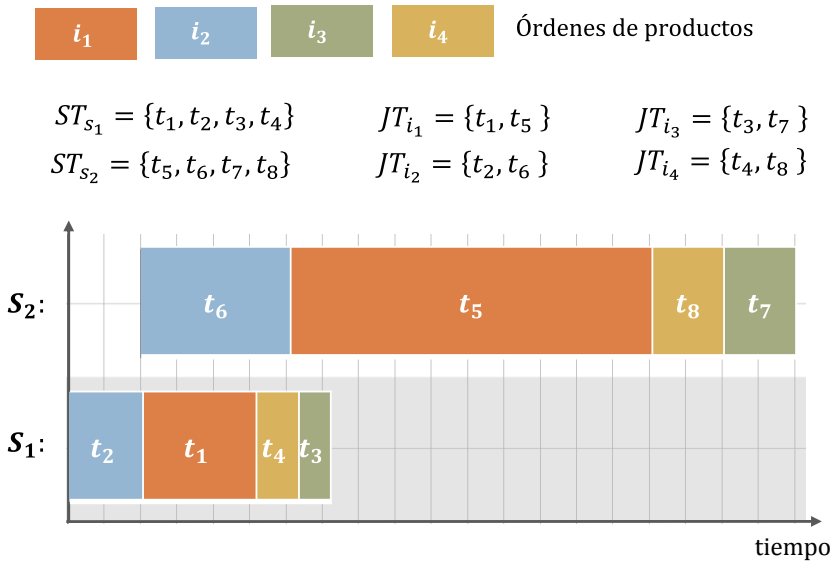



Figura 4.2. Definición de los subconjuntos $JT_i$ y $ST_s$.

## 4.3. Formulación matemática

En esta sección se presenta el modelo matemático desarrollado para resolver el problema de programación de operaciones descripto anteriormente, que considera unidades flexibles de procesamiento y restricciones de rediseño. Dicho enfoque se basa en una formulación matemática MILP con dominio continuo del tiempo, que utiliza la noción de precedencia generalizada, introducida por Méndez et al. (2001). La formulación original propuesta por estos autores se amplía para incorporar unidades de usos múltiples y modelar, al igual que los capítulos anteriores, plantas que llevan a cabo tareas de ensamblado.

Como se ha mencionado en la sección anterior, el objetivo principal del problema es determinar la operación óptima de la planta con el fin de minimizar el makespan y mejorar el uso de los equipos, mientras se cumplen todas las restricciones operativas del proceso. Por lo tanto, el modelo matemático permite determinar:

i. Asignación y secuenciación óptima de las tareas en cada máquina o unidad, de acuerdo a las recetas de procesamiento de cada producto.

ii. Determinación de los tiempos de inicio y finalización de cada tarea.

iii. Número mínimo de unidades de procesamiento necesarias para satisfacer el mejor makespan encontrado para el problema de "scheduling".

iv. Reubicación factible de unidades de procesamiento para aumentar la eficiencia del proceso.

A continuación, se definen las variables de decisión que se utilizan en el modelo, tanto para la programación como para la determinación del rediseño del sistema, y los conjuntos de restricciones que componen la formulación matemática:

- $Y_{tk}$: variable binaria que determina la asignación de las tareas $t$ a las unidades de procesamiento $k$.

- $W_{tt'}$: variable binaria que establece la secuencia relativa de cualquier par de tareas $(t, t')$ asignadas a una misma unidad de procesamiento $k$.

- $Ts_t$: variable continua positiva que define el tiempo de inicio de las actividades de procesamiento de cada tarea $t$.

- $Tf_t$: variable continua positiva que computa el tiempo de finalización del procesamiento de la tarea $t$.

- $V_k$: variable continua positiva que toma valor mayor a cero cuando la unidad de procesamiento $k$ se utiliza para realizar las operaciones de alguna etapa productiva, de lo contrario toma valor igual a cero.

- $X_{ku}$: variable continua positiva que indica si la unidad de procesamiento $k$ se reubica en la estación de trabajo $u \in U_k$ de la planta productiva.

- $X_{ks}$: variable continua positiva que establece si la unidad de procesamiento $k$ reubicada lleva a cabo las operaciones correspondientes a la etapa $s$.

### 4.3.1. Restricciones para la programación de las operaciones

**Restricciones de asignación.** Se define la restricción (4.1) para establecer la asignación de las tareas $t \in T$ a las etapas de procesamiento a través del uso de la variable binaria $Y_{tk}$, que toma valor 1 si la tarea $t \in ST_s$ se lleva a cabo en la unidad de procesamiento $k \in K_s$ para realizar la etapa $s \in S$. De este modo, la restricción asegura que cada tarea $t$ se debe procesar en una única unidad $k \in K_s$, siempre y cuando $t \in ST_s$.

$$\sum_{k \in K_s} Y_{tk} = 1 \qquad \forall\ s \in S, t \in ST_s \tag{4.1}$$

**Restricciones de tiempo**. La restricción (4.2) calcula el tiempo de finalización de la tarea $t$ $(Tf_t)$ como el tiempo de inicio $(Ts_t)$, más el tiempo de procesamiento, el cual se conoce a priori a través del parámetro $tp_t$. Si $t \in (JT_i \cap ST_s)$, entonce $tp_t = tp_{is}$.

$$Tf_t \geq Ts_t + tp_t \qquad \forall\ t \in T \tag{4.2}$$

Por otro lado, la restricción (4.3) restringe el procesamiento del producto $i \in I$ en la etapa $s \in S_i$, representado a través de la tarea $t \in (JT_i \cap ST_s)$, ya que este no puede comenzar a procesarse hasta no haber finalizado el procesamiento en la etapa previa $(s-1)$, indicado con la tarea $t' \in (JT_i \cap ST_{(s-1)})$. Esta restricción debe modificarse de acuerdo a la política de almacenamiento intermedia adoptada en cada caso: (i) si se adopta una política de almacenamiento UIS, entonces la restricción debe expresarse como una desigualdad, mientras que, (ii) si se utiliza un almacenamiento tipo NIS debe definirse a través de una igualdad.

$$Ts_t \geq Tf_{t'} \qquad \forall\ i \in I, s \in S, (t,t') \in JT_i, t \in ST_s, t' \in ST_{s-1}: s > 1 \tag{4.3}$$

**Restricciones de secuenciación**. Para secuenciar cualquier par de tareas $(t,t')$ asignadas a una misma unidad de procesamiento $k$, se definen las restricciones (4.4) y (4.5). Notar que, debido a que una unidad $k$ puede ser multipropósito, la misma puede procesar tareas asociadas a diferentes etapas productivas $s \in S$, siempre que se cumpla que $k \in K_s$. La Figura 4.3 muestra un ejemplo de una unidad multipropósito $(k_2)$ que procesa 5 tareas ($t_5$-$t_9$), correspondientes al procesamiento de 4 productos ($i_1$-$i_4$) en 2 etapas diferentes ($s_2$ y $s_3$). Por ejemplo, las tareas $t_5$ y $t_9$ representan el procesamiento del producto $i_1$ en la etapa $s_2$ y $s_3$, respectivamente.

$$Ts_{t'} \geq Tf_t - M(1 - W_{tt'}) - M(2 - Y_{tk} - Y_{t'k}) \tag{4.4}$$
$$\forall\ (s,s') \in S, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}): t < t'$$

$$Ts_t \geq Tf_{t'} - MW_{tt'} - M(2 - Y_{tk} - Y_{t'k}) \tag{4.5}$$
$$\forall\ (s,s') \in S, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}): t < t'$$

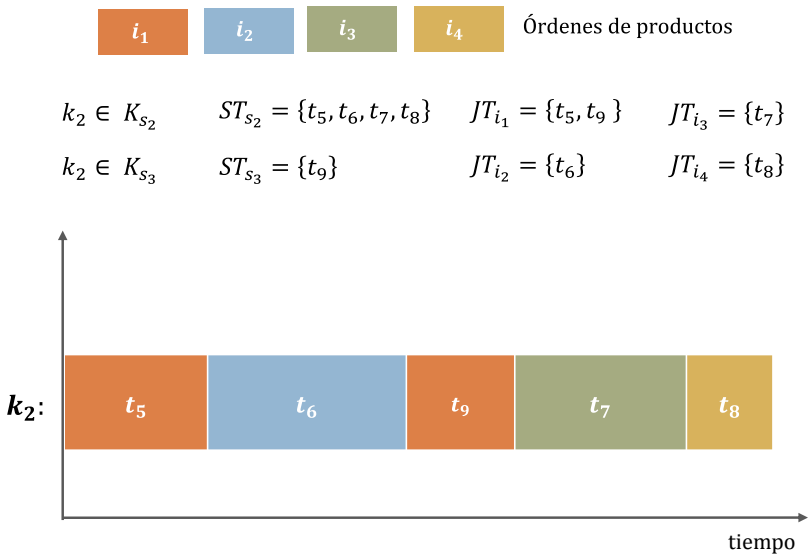



Figura 4.3. Unidad multipropósito que realiza las operaciones de dos etapas de procesamiento.

Siguiendo el concepto de precedencia general, las restricciones de secuenciación en una unidad $k$ se definen para cualquier par de tareas $(t, t')$, donde $t < t', t \in ST_s, t' \in ST_{s'}$ y $k \in (K_s \cap K_{s'})$. Para modelar dichas restricciones, se utiliza la variable binaria de secuenciación $W_{tt'}$, la cual toma valor 1 cuando $t$ se procesa antes que $t'$, en caso contrario toma valor 0. Si dos tareas $(t, t')$ se asignan a una misma unidad $k$ ($Y_{tk} + Y_{t'k} = 2$) y la tarea $t$ se procesa antes que la tarea $t'$ ($W_{tt'} = 1$), la restricción (4.4) fuerza que el tiempo de inicio de $t', Ts_{t'}$, sea mayor que el tiempo de finalización de la tarea $t, Tf_t$. Caso contrario, si la tarea $t'$ se procesa antes que $t$, es decir, $W_{tt'} = 0$, la desigualdad (4.4) se vuelve redundante y la restricción (4.5) se activa para forzar la secuencia.

Por otro lado, puede resultar conveniente secuenciar las tareas sin la utilización de la variable binaria $W_{tt'}$. La restricción (4.6) refleja esta situación, donde la secuencia de procesamiento de cualquier par de tareas $(t, t')$, con $t < t'$, en una determinada unidad $k$ queda establecida por el orden en que dichas tareas son definidas en el conjunto $T$. Se toma como hipótesis que las tareas $(t, t')$, asociadas al procesamiento de un mismo producto en dos etapas diferentes, $s$ y $s'$ tal que $s < s', t \in ST_s$ y $t' \in ST_{s'}$, siguen el orden: $t < t'$. De este modo, si las dos tareas son asignadas a la unidad $k$ ($Y_{tk} = Y_{t'k} = 1$),

entonces siempre se van a procesar en orden creciente, independientemente de la etapa $s$ a la que estén asociadas.

$$Ts_{t'} \geq Tf_t - M(2 - Y_{tk} - Y_{t'k}) \quad (4.6)$$

$$\forall\, (s, s') \in S, t \in ST_s, t' \in ST_{s'}, k \in (K_s \cap K_{s'}): t < t'$$

**Restricciones de ensamblado.** Por otra parte, con el fin de modelar posibles operaciones de ensamblado se definen las restricciones (4.7)-(4.9). La primera establece que el procesamiento del producto $i$ en la etapa de ensamblado $s \in S^a$ no puede comenzar hasta que sus respectivos subensambles $i' \in SA_i$ hayan completado su procesamiento en la etapa previa $(s-1)$, mientras que las dos últimas modelan secuencias de ensamblado, definidas por el parámetro $Seq_i$, que se deben satisfacer en las etapas $s \in S^a_{seq}$, a través de las variables continuas de tiempo (4.8) y de las variables binarias de secuenciación (4.9).

$$Ts_t \geq Tf_{t'} \qquad \forall\, s \in S^a, i \in I^a, i' \in SA_i, t \in JT_i, t' \in JT_{i'}, t \in ST_s, t' \in ST_{s-1} \quad (4.7)$$

$$TF_t \geq TF_{t'} \quad (4.8)$$

$$\forall\, (i, i') \in I^a, t \in JT_i, t' \in JT_{i'}, s \in S^a_{seq}, s \in (S_i \cap S_{i'}), (t, t') \in ST_s: Seq_i < Seq_{i'}$$

$$W_{tt'} = 1 \quad (4.9)$$

$$\forall\, (i, i') \in I^a, t \in JT_i, t' \in JT_{i'}, s \in S^a_{seq}, s \in (S_i \cap S_{i'}), (t, t') \in ST_s: Seq_i < Seq_{i'}$$

**Función objetivo.** La función objetivo para la formulación MILP es minimizar el makespan ($MK$) y está representada por la restricción (4.10). En este caso, el tiempo mínimo de finalización de todas las tareas $t \in T$ se calcula mediante la restricción (4.11).

$$minimize\ MK \quad (4.10)$$

$$MK \geq Tf_t \qquad \forall\ t \in T \tag{4.11}$$

### 4.3.2. Restricciones para el rediseño de la planta

Una vez encontrada la mejor solución al problema de programación de operaciones, es decir, una vez resuelto el modelo que considera las restricciones (4.1)-(4.5) y (4.7)-(4.11), se define un nuevo objetivo para minimizar el número de unidades o equipos utilizados en cada estación de trabajo $u \in U$. De este modo, la solución actual para el problema de "scheduling" detallado, se establece como cota superior para la variable $MK$ y se resuelve nuevamente el modelo matemático, reemplazando la función objetivo (4.10) por la expresión (4.12) y considerando en esta ocasión la restricción (4.13).

$$minimize \sum_{k \in K} V_k \tag{4.12}$$

$$Y_{tk} \leq V_k \qquad \forall\ t \in T, k \in K \tag{4.13}$$

A través de la restricción (4.13), la variable continua $V_k$ toma un valor mayor a 1 siempre que alguna tarea $t \in T$ sea asignada para ser procesada en la unidad $k \in K$, es decir, $Y_{tk} = 1$. Sin embargo, como la variable $V_k$ se utiliza dentro de un término positivo en la función objetivo de minimización (4.12), el modelo le asigna a dicha variable sólo valores 0 y 1, lo cual refleja un comportamiento como variable binaria. De esta manera, cuando $V_k$ toma valor igual a 0, se considera que la unidad de procesamiento $k$ no se utiliza para realizar operaciones de procesamiento, y por lo tanto, se considera que dicha unidad "se libera" y pasa a formar parte del conjunto $L$. En otras palabras, se deja de utilizar la unidad $k$ para realizar las operaciones originalmente asignadas, para posteriormente evaluar su reubicación a otra "workstation" compatible $u \in U_k$.

**Reubicación de las unidades liberadas a estaciones de trabajo.** Con el fin de mejorar la eficiencia del proceso productivo, y por lo tanto, el valor del makespan, se lleva a cabo la evaluación de posibles reconfiguraciones de la planta, considerando la configuración actual, las etapas cuello de botellas y las reubicaciones permitidas de las unidades de procesamiento "liberadas" anteriormente. De este modo, se modela la reasignación tanto, de unidades $k \in L$ a estaciones de trabajo $u \in U_k$, como de tareas $t \in T$ a dichas unidades.

En particular, cada unidad $k \in L$ se puede reubicar en una única "workstation" $u \in U_k$, la cual debe ser compatible con dicha unidad. Por consiguiente, se define la restricción (4.14), que establece que una unidad liberada puede reubicarse a lo sumo en una estación de trabajo. Asimismo, en caso que se decida llevar a cabo la reubicación, es decir, $X_{ku} = 1$, y debido a las unidades pueden ser multipropósito, la unidad de procesamiento $k$ puede procesar las tareas correspondientes a todas las etapas productivas $s$ que se llevan a cabo en la estación $u$: $s \in S_u$. Esta situación se modela a través de la restricción (4.15).

$$\sum_{u \in U_k} X_{ku} \leq 1 \qquad \forall\ k \in L \tag{4.14}$$

$$X_{ku} = X_{ks} \qquad \forall\ u \in U_k, k \in L, s \in S_u \tag{4.15}$$

**Reprogramación de tareas a unidades reubicadas.** Cuando se incorpora una unidad $k \in L$ a una estación de trabajo $u \in U_k$, las tareas $t$ que se procesaban en dicha estación, pueden ser reasignadas a la nueva unidad. En este caso, se utiliza la variable binaria $Y_{tk}$ para computar la asignación de tareas $t$ sólo en las unidades de procesamiento liberadas $k \in L$. En primer lugar, se fija dicha variable en cero para aquellas unidades no liberadas ($k \notin L$), dejándola libre para las $k$ con posibilidad de reubicación. De este modo, por medio de la restricción (4.16), la variable $Y_{tk}$ puede tomar valor igual a 1 cuando a una unidad reubicada se le asigna una o más tareas. Además, si la unidad $k \in L$ se reubica en

alguna estación $u \in U_k$, se cumple que $X_{ku} = 1$, y como se mencionó anteriormente, $X_{ks} = 1$ para todas las etapas de procesamiento $s \in S_u$ de dicha estación. De este modo, a través de la restricción (4.17), las tareas $t$ que se llevan a cabo en $s$ ($t \in ST_s$) pueden asignarse a la nueva unidad ($Y_{tk} = 1$).

$$\sum_{k \in L} Y_{tk} \leq 1 \qquad \forall\ t \in T \tag{4.16}$$

$$Y_{tk} \leq X_{ks} \qquad \forall\ s \in S, t \in ST_s, k \in L \tag{4.17}$$

**Ajuste de los tiempos de inicio de procesamiento y de secuenciaciones.** Si una unidad $k$ está en uso, es decir no fue liberada ($k \notin L$), y tiene asignadas originalmente las tareas $t$ y $t'$, tal que $t$ precede a $t'$ ($W_{tt'} = 1$), pero la tarea $t$ es reasignada a una unidad nueva $k' \in L$, entonces se deben actualizar las respectivas decisiones de tiempo, modeladas a través de las restricciones (4.18) y (4.19). Asimismo, se modela las decisiones de secuenciación para las nuevas unidades $k'$ definiendo las restricciones (4.20) y (4.21).

En el primer caso, correspondiente al ajuste de decisiones de tiempos a través de las restricciones (4.18) y (4.19), se utiliza el parámetro $M$, y los valores de los parámetros correspondientes a las decisiones de asignación y secuenciación actuales: $sY_{tk}$ y $sW_{tt'}$, respectivamente. De este modo, si una tarea $t$ es reasignada a una unidad nueva $k' \in L$, entonces el tiempo de inicio de las tareas $t'$ que la precedían antes de la reasignación ($sW_{tt'} = 1$), debe ser mayor a 0, es decir, se relaja. Esto se debe satisfacer siempre que se cumpla que las tareas $t$ y $t'$ hayan estado asignadas originalmente a la misma unidad $k \notin L$: $sY_{tk} = sY_{t'k} = 1$. Sin embargo, si no se lleva a cabo la reasignación de dicha tarea $t$, se debe seguir cumpliendo la restricción que establece que el tiempo de inicio de las tareas predecesoras $t'$ debe ser mayor al tiempo de finalización de la tarea antecesora $t$.

$$Ts_{t'} \geq Tf_t - M\left(\sum_{k' \in L} Y_{tk'} + \sum_{k' \in L} Y_{tk'}\right) \qquad (4.18)$$

$$\forall\, (t,t') \in T, k \notin L : t < t', sW_{tt'} = 1, sY_{tk} = 1, sY_{t'k} = 1$$

$$Ts_t \geq Tf_{t'} - M\left(\sum_{k' \in L} Y_{tk'} + \sum_{k' \in L} Y_{tk'}\right) \qquad (4.19)$$

$$\forall\, (t,t') \in T, k \notin L : t < t', sW_{tt'} = 0, sY_{tk} = 1, sY_{t'k} = 1$$

Por otro lado, en cuanto a las decisiones de secuenciación, se deben tener en cuenta las tareas reasignadas a cada unidad nueva $k \in L$. De manera similar a las restricciones (4.4) y (4.5) antes definidas, se establecen las expresiones (4.20) y (4.21) para cualquier par de tareas $(t, t')$, tal que $t < t'$. Si dichas tareas se asignan a una unidad $k \in L$ previamente liberada, es decir $Y_{tk} = 1$ y $Y_{t'k} = 1$, y la tarea $t$ precede a la tarea $t'$ en dicha unidad ($W_{tt'} = 1$), se debe cumplir que el tiempo de inicio de procesamiento de $t'$ ($Ts_{t'}$) sea mayor que el tiempo de finalización de la tarea $t$ ($Tf_t$). Esto se modela a través de la restricción (4.20). En el caso que la tarea $t'$ preceda a la tarea $t$ ($W_{tt'} = 0$), se activa la restricción (4.21), forzando la secuencia.

$$Ts_{t'} \geq Tf_t - M(1 - W_{tt'}) - M(2 - Y_{tk} - Y_{t'k}) \qquad \forall (t,t') \in T, k \in L: t < t' \qquad (4.20)$$

$$Ts_t \geq Tf_{t'} - MW_{tt'} - M(2 - Y_{tk} - Y_{t'k}) \qquad \forall (t,t') \in T, k \in L: t < t' \qquad (4.21)$$

En la Figura 4.4 se ilustra un ejemplo, donde se puede observar la reubicación de una unidad ($k_2$) a una estación de trabajo compatible ($u_1$) y por lo tanto, la reasignación de tareas a dicha unidad: $t_2$ $y$ $t_3$. Por consiguiente, los tiempos de inicio y finalización de procesamiento de las 4 tareas involucradas se deben actualizar a partir de las restricciones antes descriptas.

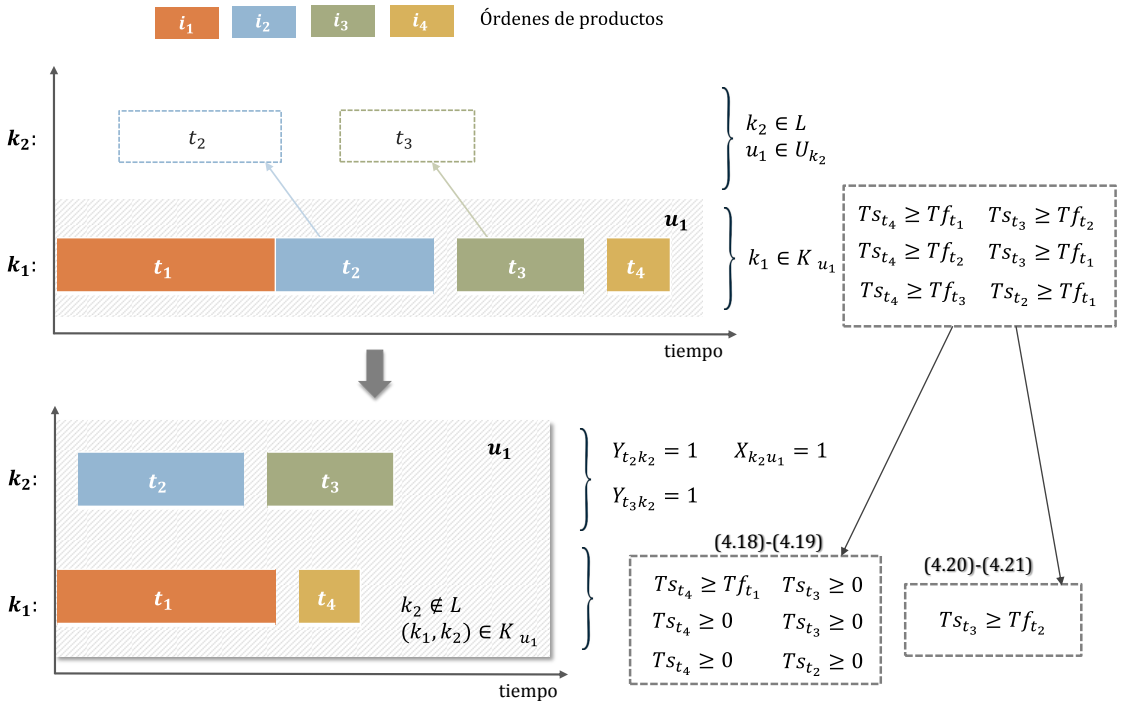



Figura 4.4. Reasignación de tareas a unidades nuevas y ajuste de tiempos.

## 4.4. Metodología integrada de programación de las operaciones y rediseño

Como se ha mencionado anteriormente, el problema abordado en este capítulo considera, tanto las decisiones de programación de las operaciones, como las de rediseño de plantas tipo job shop flexibles, por lo que se considera un problema NP-hard con una elevada complejidad combinatoria (Pinedo, 2016). Como se ha demostrado en los capítulos anteriores de esta tesis, un enfoque exacto de espacio completo, como el presentado en la sección anterior, no es el más conveniente para resolver problemas industriales de gran escala. Razón por la cual, se adopta una metodología basada en un algoritmo que resuelve de manera iterativa el modelo MILP propuesto.

Considerando la estrategia de solución basada en técnicas de descomposición desarrollada en el Capítulo 3, se propone un nuevo algoritmo superador que presenta una mayor flexibilidad en las decisiones de asignación y secuenciación, e incorpora una etapa de evaluación de utilización de recursos para el rediseño de la configuración de una planta

tipo job shop flexible, a fin de reportar una solución de alta calidad con mínimo esfuerzo computacional. De este modo, la estrategia incorpora nuevas líneas de código en un intento de aumentar la flexibilidad del algoritmo y determinar el mejor programa de operaciones y configuración de la planta, considerando unidades multipropósito, operaciones de ensamblado y restricciones de rediseño, al mismo tiempo que se minimiza el makespan.

El esquema general del nuevo algoritmo se muestra en la Figura 4.5, donde se pueden observar dos etapas principales, correspondientes a los problemas de "scheduling" y rediseño, respectivamente.

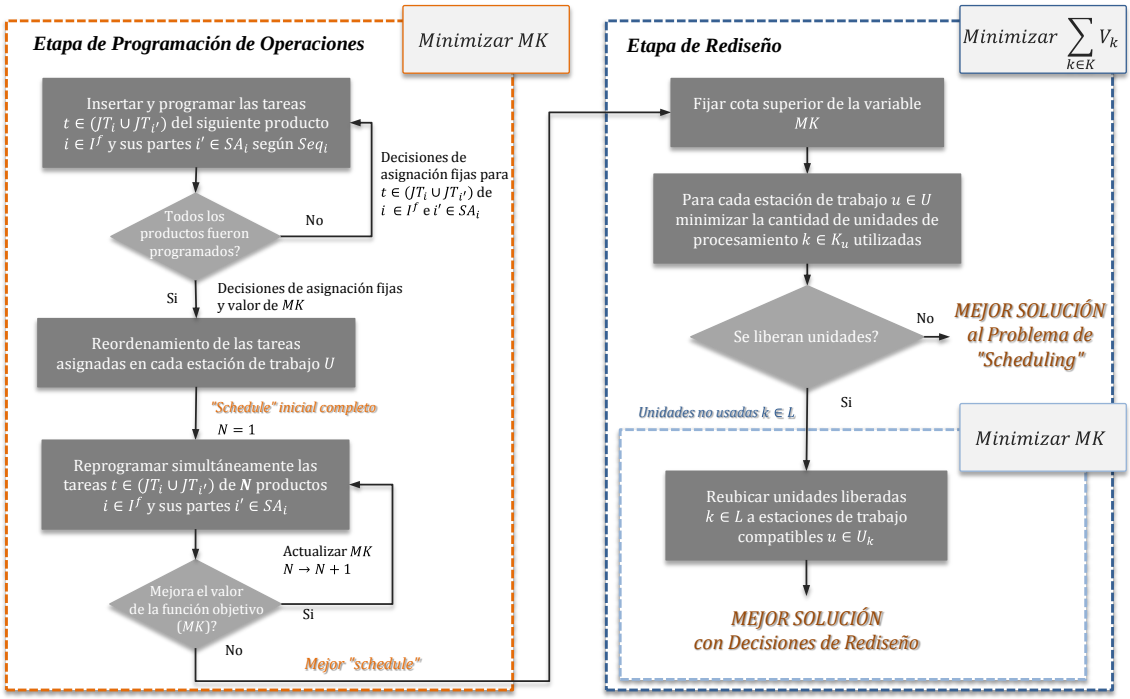



Figura 4.5. Esquema general del algoritmo iterativo de programación y rediseño.

(i) *Etapa de programación de operaciones*: se establece una solución inicial, que luego es mejorada de manera gradual, considerando la reasignación y reordenamiento de tareas con el fin de obtener el mejor "schedule".

(ii) *Etapa de rediseño*: se evalúa la utilización de las unidades multipropósito en cada estación de trabajo, determinando posibles sobredimensionamientos de

las mismas y analizando la reducción del uso de dichas unidades, al mismo tiempo que se tiene en cuenta el mejor "schedule" encontrado en la etapa de programación. Además, se estudian posibles reubicaciones factibles de unidades de procesamiento a otras estaciones de trabajo, para llevar a cabo las operaciones de las etapas productivas involucradas en las mismas y de este modo, mejorar la eficiencia del sistema productivo y el "schedule".

### 4.4.1. Etapa de programación detallada de las operaciones

La primera etapa del procedimiento iterativo tiene como objetivo determinar el mejor programa de operaciones de manufactura para el problema bajo estudio. Esta primera parte del procedimiento está integrada por dos sub-etapas, las cuales se llevan a cabo de manera secuencial.

#### *Solución inicial factible*

Se define un procedimiento iterativo para encontrar un programa de operaciones factible considerando el conjunto completo de productos $i \in I$, la secuencia de procesamiento de cada uno y las unidades de procesamiento disponibles $k \in K$. La búsqueda de una solución factible se lleva a cabo usando la heurística de construcción secuencial propuesta en el Capítulo 3 (sección 3.3.1) y aplicando una fase posterior de reordenamiento de tareas asignadas a unidades de procesamiento. De este modo, se programa un subconjunto de órdenes de productos, hasta que finalmente se insertan todos los productos $i \in I$ y se obtiene un "schedule" factible.

En esta primera sub-etapa de programación de operaciones es necesario establecer un criterio para seleccionar la cantidad de productos y el orden en que deben ser insertados en cada iteración del algoritmo. Al igual que el capítulo anterior, se propone insertar uno a uno los productos finales $i \in I^f$, de acuerdo a una secuencia de inserción preestablecida bajo algún criterio específico, la cual queda definida por el parámetro $Seq_i$. Debido a que los productos finales $i \in I^f$ pueden estar conformados por subensambles

$i' \in SA_i$, estos deben ser insertados y programados de manera conjunta. De esta forma, en cada iteración, se asignan y secuencian todas las tareas asociadas a un producto final $i$ y sus respectivos subensambles $i'$: $t \in (JT_i \cap JT_{i'})$. El pseudocódigo de esta primera sub-etapa del problema de "scheduling" se muestra en la Figura 4.6.

---

Set $iter = 1, active_t = false$;
WHILE $iter \leq |I^f|$
  LOOP ($i \in I^f$ and $iter = Seq_i$)
    LOOP ($t \in JT_i$)
      $active_t = true;$
    END LOOP;
    LOOP ($i' \in SA_i$)
      LOOP ($t \in JT_{i'}$)
        $active_t = true;$
      END LOOP;
    END LOOP;
  END LOOP;
  Resolver Modelo MILP (4.1)-(4.3), (4.6)-(4.8), (4.10)-(4.11);
  LOOP ($t \in T$ and $active_t = true$)
    Fijar variables binarias $Y_{tk}$;
  END LOOP;
  $iter = iter + 1;$
END WHILE
$sMK = MK$;
LOOP ($t \in T$)
  LOOP ($k \in K$)
    $sY_{tk} = Y_{tk}$;
  END LOOP;
END LOOP;
Resolver Modelo MILP (4.1)-(4.5),(4.7)-(4.11);
LOOP ($(t, t') \in T$ and $t < t'$)
  $sW_{tt'} = W_{tt'}$;
END LOOP;
Liberar variables binarias $Y_{tk}$;

---

Figura 4.6. Pseudocódigo - Solución inicial factible.

Al igual que el algoritmo utilizado en la sección 3.3.1 del Capítulo 3, se usa un parámetro booleano llamado $active_t$ para determinar si la tarea $t$ debe ser asignada y secuenciada en la iteración $iter$. Notar también que, el algoritmo realiza tantas iteraciones como productos finales haya que insertar y programar, por lo tanto, el parámetro $iter$ va a tomar valores entre 1 y $|I^f|$. Después de cada ejecución del modelo matemático, definido por las restricciones (4.1)-(4.3), (4.6)-(4.8), (4.10) y (4.11), las variables binarias de asignación $Y_{tk}$ son fijadas para aquellas tareas $t$ con $active_t = true$.

Cabe destacar que, una diferencia con la "etapa de construcción" definida en la sección 3.3.1, es que en este caso, se realiza un reordenamiento de todas las tareas $t \in T$ que se llevan a cabo en cada estación de trabajo $u \in U$. Esto se realiza una vez definidas y fijadas todas las decisiones de asignación de tareas a unidades de procesamiento, a través de la variable binaria $Y_{tk}$. De este modo, se procede a resolver el modelo MILP correspondiente a las restricciones (4.1)-(4.5) y (4.7)-(4.11). En particular, se incluyen las restricciones que utilizan la variable binaria $W_{tt'}$, para determinar las decisiones de secuenciación de las tareas asignadas a cada unidad de procesamiento $k \in K$.

Por último, el algoritmo utiliza los parámetros $sMK$, $sY_{tk}$, y $sW_{tt'}$ para guardar el valor de la función objetivo y las decisiones de asignación y secuenciación, respectivamente, correspondientes a la solución inicial encontrada.

### *Mejora gradual de la solución del problema de programación*

En esta fase del algoritmo, la solución inicial reportada se mejora realizando iteraciones de reprogramación. Cada acción de reprogramación consiste en ejecutar el modelo MILP completo, pero sólo considerando un subconjunto de productos $i \in I$ del programa de operaciones actual, con el fin de evaluar posibles reasignaciones y reordenamientos que permitan mejorar el valor de la función objetivo ($MK$).

Es importante destacar que, cuando un producto final $i \in I^f$ es liberado, da lugar a que todas las tareas asociadas al mismo ($t \in JT_i$) pueden reprogramarse. Del mismo

modo, ocurre con las tareas asociadas a los subensambles de dicho producto $t' \in JT_{i'}$, tal que $i' \in SA_i$. En cada ejecución del modelo, el parámetro $active_t$ toma valor verdadero para todas las tareas que pueden ser reasignadas y/o reordenadas. Por otro lado, para todas las tareas con $active_t = false$, sólo se pueden tomar decisiones de tiempo.

Un punto clave a tener en cuenta en esta fase del algoritmo es la cantidad de productos $i \in I^f$ a reprogramar simultáneamente en cada ejecución del modelo, es decir, en cada iteración de mejora. Este valor está representado por el parámetro $N$. Para ampliar su explicación se considera un pequeño problema de "scheduling", donde se deben procesar 4 productos ($i_1, i_2, i_3, i_4$). La Figura 4.7 muestra los productos liberados y reprogramados en cada iteración, para los diferentes valores que puede tomar $N$, es decir, desde $N = 1$ a $N = |I|$. Como se puede observar, cuanto menor es el valor de $N$, mayor es el número de iteraciones y menor el espacio de búsqueda del "solver" MIP, lo que da como resultado tamaños manejables del modelo. Sin embargo, a medida que aumenta el valor de $N$: (i) se reduce el número de iteraciones, (ii) aumenta la flexibilidad de la reprogramación, (iii) la región factible resulta mayor en cada una de ellas y, por lo tanto, (iv) las formulaciones matemáticas resultantes pueden volverse más complejas, hasta intratables en algunos casos.

| | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ |
|---|---|---|---|---|
| $iter = 1$ | $i_1$ | $i_1$ $i_2$ | $i_1$ $i_2$ $i_3$ | $i_1$ $i_2$ $i_3$ $i_4$ |
| $iter = 2$ | $i_2$ | $i_2$ $i_3$ | $i_2$ $i_3$ $i_4$ | |
| $iter = 3$ | $i_3$ | $i_3$ $i_4$ | | |
| $iter = 4$ | $i_4$ | | | |

Figura 4.7. Productos reprogramados en cada iteración para diferentes valores de $N$.

Por lo expuesto, se propone ejecutar la mejora iterativa comenzando con $N = 1$ e ir incrementando este valor, de uno en uno, hasta que la resolución de cada modelo MILP, definido por las restricciones (4.1)-(4.5) y (4.7)-(4.11), no mejore la solución actual dentro de un límite de CPU predefinido (por ejemplo, 120 CPUs), el cual está representado por el

parámetro $ltime$. Asimismo, se debe fijar un valor máximo $N^{max}$ de productos a reprogramar de manera simultánea, cuyo valor se puede obtener a partir de un análisis previo del problema. El pseudocódigo de esta fase se muestra en la Figura 4.8.

Set $N = 1$; $bestSol = sMK$ ; $control = true$;
WHILE $N <$ $\boldsymbol{N^{max}}$
  WHILE $control$
    $control = false$;
    $iter = 1$;
    WHILE $iter \leq (|I^f| - N + 1)$
      $active_t = false$;
      LOOP ($i \in I^f$ and $i = iter$)
        LOOP ($t \in JT_i$)
          $active_t = true$;
        END LOOP;
        LOOP ($i' \in SA_i$)
          LOOP ($t \in JT_{i'}$)
            $active_t = true$;
          END LOOP;
        END LOOP;
      END LOOP;
      Set $\boldsymbol{lTime}$
      Resolver Modelo MILP (4.1)-(4.5), (4.7)-(4.11)
      IF ($MK <$ $bestSol$)
        $bestSol = MK$;
        $control = true$;
        LOOP ($t \in T$ and $active_t = true$)
          $sY_{tk} = Y_{tk}$;
        END LOOP;
        LOOP ($(t, t') \in T$ and $t < t'$ and
          ($active_t = true$ or $active_{t'} = true$))
          $sW_{tt'} = W_{tt'}$;
        END LOOP;
      END IF;
      $iter = iter + 1$;
    END WHILE;
  END WHILE;
  $N = N + 1$;
END WHILE

Figura 4.8. Pseudocódigo – Mejora iterativa.

Luego de cada ejecución del modelo MILP, se verifica el valor de la función objetivo ($MK$). Si este es mejor que el último reportado, es decir, el "schedule" encontrado permite minimizar aún más el valor $MK$, se actualiza el mejor makespan encontrado ($BestSol$) y se guardan las nuevas decisiones de asignación y secuenciación, actualizando los valores de los parámetros $sY_{tk}$, y $sW_{tt'}$, respectivamente. Una vez que el algoritmo evalúa la reprogramación de las tareas asociadas a todos los productos $i \in I^f$ e $i' \in SA_i$, verifica nuevamente si el valor de la función objetivo ha mejorado a través del parámetro booleano $control$, el cual se activa (tomando valor verdadero) cuando se produce una mejora. En este caso, el algoritmo comienza nuevamente las iteraciones, reprogramando las tareas desde el principio. De lo contrario, el procedimiento algorítmico termina y reporta el programa de operaciones actual como la mejor solución encontrada.

### 4.4.2. Etapa de rediseño

La última etapa del algoritmo evalúa el rediseño de la planta, donde a partir del mejor "schedule" obtenido por el problema de programación de operaciones: (i) se analiza si la planta de producción está sobredimensionada en alguna celda de trabajo y (ii) se evalúan reubicaciones factibles de unidades a otras celdas, de manera de mejorar el "schedule" y la utilización de los equipos. Al igual que la primera etapa, se llevan a cabo dos sub-etapas secuencialmente.

#### *Análisis de la utilización de las unidades de procesamiento*

En la primera fase de la etapa de rediseño se lleva a cabo un análisis de la utilización de los equipos o unidades de procesamiento que componen cada estación de trabajo. El objetivo principal es detectar si alguna "workstation" se encuentra sobredimensionada y determinar la cantidad mínima necesaria de unidades de procesamiento $k \in K_u$ requeridas por las mismas para procesar todos los productos asignados, mientras se cumple el valor de la función objetivo reportado como la mejor solución de la etapa de programación.

De este modo, una vez obtenido el mejor programa de operaciones que, en este caso minimiza el makespan, el primer paso de la presente fase de la metodología iterativa es fijar el valor de la variable $MK$. Luego, se ejecutan iteraciones de reprogramación, donde se resuelve el modelo MILP, que tiene en cuenta las restricciones (4.1)-(4.5), (4.7)-(4.9) y (4.11)-(4.13), para cada estación de trabajo $u \in U$.

El pseudocódigo para la primera sub-etapa de rediseño se muestra en la Figura 4.9. Notar que, en cada iteración, el algoritmo activa todas las tareas ejecutadas en cada estación $u \in U$, con el fin de poder reprogramarlas y minimizar la cantidad de unidades $k \in K_u$ requeridas.

---

Set $iterU = 1$
Fijar valor de la variable $MK = BestSol$
WHILE $iterU \leq |U|$
  $active_t = false$;
  LOOP ($s \in S_u$ and $u = iterU$)
    LOOP ($t \in ST_s$)
      $active_t = true$;
    END LOOP;
  END LOOP;
  Resolver Modelo MILP (4.1)-(4.5), (4.7)-(4.9), (4.11)-(4.13)
  LOOP ($k \in K_u$ and $u = iterU$)
    Fijar variables positivas $V_k$;
    LOOP ($t \in T$ and $active_t = true$)
      $sY_{tk} = Y_{tk}$;
    END LOOP;
  END LOOP;
  LOOP $((t, t') \in T$ and $t < t'$and ($active_t = true$
    or $active_{t'} = true$))
    $sW_{tt'} = W_{tt'}$;
  END LOOP;
  $iterU = iterU + 1$;
END WHILE

---

Figura 4.9. Pseudocódigo – minimización de unidades de procesamiento.

Específicamente, en cada iteración, se establece el valor del parámetro $active_t$ en verdadero para activar todas las tareas $t \in ST_s$, realizadas en las etapas productivas $s \in S_u$ que se pueden llevar a cabo en cada estación $u$, de manera de poder reasignar dichas tareas a otras unidades $k \in K_u$ de la misma estación. De este modo, el procedimiento de descomposición implementado permite reducir el espacio de búsqueda del "solver" MIP en cada iteración.

Luego de cada ejecución del modelo MILP, se fija el valor de la variable $V_k$ para todas las unidades $k$ involucradas en la iteración actual. Dicha variable toma valor 1 si se utiliza $k$ en la estación $u$ ($k \in K_u$), en otro caso, toma valor 0, lo cual refleja que la unidad $k$ "se libera" para una posterior reubicación. En este último caso, dicha unidad pasa a formar parte del conjunto $L$ ($k \in L$). Asimismo, se guardan las decisiones de asignación y secuenciación al finalizar cada ejecución del modelo matemático, en los parámetros $sY_{tk}$ y $sW_{tt'}$, respectivamente. El procedimiento finaliza cuando todas las estaciones de trabajo $u \in U$ fueron evaluadas.

### *Reconfiguración del sistema de manufactura flexible*

Una vez identificadas las unidades de procesamiento liberadas ($k \in L$) en la primera fase de la etapa de rediseño, se procede a evaluar la reubicación de las mismas a otras posibles estaciones de trabajo $u \in U_k$. Las reasignaciones factibles se definen para cada problema en particular, estableciendo las estaciones de trabajo compatibles de cada unidad a través del conjunto $U_k$, previamente definido en este capítulo. En principio, se puede suponer que las unidades liberas $k \in L$ pueden ser reubicadas en cualquier "workstation" $u \in U$.

Es importante destacar que, el objetivo de esta última fase del algoritmo, es analizar la configuración del sistema productivo y llevar a cabo una reubicación eficiente de todas las unidades liberadas previamente, de manera de minimizar el makespan y mejorar la utilización de las estaciones de trabajo. Por lo tanto, al igual que la primera fase,

se debe definir como cota superior de la variable $MK$, el mejor valor reportado en la etapa de programación de las operaciones de la metodología iterativa propuesta ($BestSol$).

Se presentan dos posibles heurísticas para modelar las decisiones de rediseño de la planta. Cabe destacar que, en ambas estrategias se ejecuta el modelo MILP conformado por las restricciones (4.2)-(4.3), (4.7)-(4.11) y (4.14)-(4.21). A continuación, se describen los enfoques propuestos.

- *Heurística 1:* Las unidades liberadas $k \in L$ en la primera sub-etapa de rediseño son reubicadas de manera iterativa en alguna estación de trabajo $u \in U_k$ compatible, de una en una, evaluando en cada ejecución todas las posibles reubicaciones posibles. En este caso, el pseudocódigo utilizado se detalla en la Figura 4.10.

  Como se puede observar, en cada iteración se activan las decisiones de asignación y secuenciación para cada unidad $k \in L$ en aquellas celdas $u \in U_k$, a través del parámetro booleano $active_k$. Notar que, se libera la variable binaria $Y_{tk}$, para tomar valor igual a 1 si la tarea $t$ es reasignada a una unidad nueva y 0 en caso contrario. Además, se liberan las variables positivas $X_{ks}$ y $X_{ku}$, que denotan la utilización de la unidad de procesamiento $k$ previamente liberada, tanto en la estación de trabajo $u$, como para efectuar las operaciones de las etapas $s \in S_u$ de dicha estación.

  Luego de cada ejecución del modelo MILP, una unidad es reubicada de manera de minimizar el makespan. De este modo, se actualizan los parámetros $sY_{tk}$ y $sW_{tt'}$ con la nueva solución reportada. Esta fase algorítmica finaliza cuando todas las unidades $k \in L$ son reasignadas a alguna celda $u \in U_k$.

- *Heurística 2:* Se modela la reubicación de todas las unidades liberadas de manera simultánea, es decir, la presente heurística está basada en la resolución de la fase de reconfiguración de la planta de manera integrada. Por lo tanto, se activan todas las unidades liberadas, $active_k = true$ para $\forall k \in L$, y se resuelve el modelo MILP,

permitiendo la reubicación de las unidades y la reprogramación de las tareas en dichas unidades.

---

Set $iterUnit = 1$
Fijar cota superior de la variable $MK = BestSol$
WHILE $iterUnit \leq |L|$
  $active_k = false$;
  Liberar variables binarias $Y_{tk}$;
  Liberar variables positivas $X_{ks}$ ;
  Liberar variables positivas $X_{ku}$ ;
  LOOP ($k \in L$ and $k = iterUnit$)
    $active_k = true$;
    LOOP ($u \notin U_k$)
      Fijar variables positivas $X_{ku} = 0$;
    END LOOP;
  END LOOP;
  LOOP ($k \in L$ and $active_k = false$)
    Fijar variables binarias $Y_{tk} = 0$;
    Fijar variables positivas $X_{ks} = 0$;
  END LOOP;
  Resolver Modelo MILP (4.2)-(4.3), (4.7)-(4.11), (4.14)-(4.21)
  LOOP ($k \in K$ and $k = iterUnit$ )
    LOOP ($t \in T$)
      $sY_{tk} = Y_{tk}$;
    END LOOP;
    LOOP ($(t, t') \in T$ and $t < t'$and ($Y_{tk} = 1$ or $Y_{t'k} = 1$)
      $sW_{tt'} = W_{tt'}$;
    END LOOP;
  END LOOP;
  $iterUnit = iterUnit + 1$;
END WHILE

---

Figura 4.10. Pseudocódigo – reconfiguración iterativa.

## 4.5. Casos de estudios y discusión de resultados

A fin de testear la performance computacional y la aplicabilidad del algoritmo propuesto se presentan diferentes problemas industriales derivados de dos casos de estudio. El objetivo es evaluar e ilustrar el rendimiento y la flexibilidad de la estrategia de solución mediante la resolución de instancias de distintos tamaños de dichos casos, inherentes a diferentes sectores de la industria. Cabe mencionar que, para todos los ejemplos planteados, se considera como función objetivo la minimización del makespan.

Todos los ejemplos considerados en esta sección se resuelven utilizando los dos enfoques presentados: (i) el modelo MILP monolítico y (ii) la estrategia algorítmica de solución para la programación de las operaciones y el rediseño del sistema productivo. De este modo, el estudio experimental permite estimar una medida aproximada de la calidad de las soluciones proporcionadas y evaluar la eficiencia computacional de la estrategia de solución propuesta considerando, tanto la calidad de las soluciones, como el esfuerzo computacional requerido para alcanzarlas.

En primer lugar, se valida el algoritmo a través de la resolución de un pequeño ejemplo ilustrativo de un sistema de fabricación con operaciones de ensamblado y unidades multipropósito. Posteriormente, se exponen los dos casos de estudios de tamaño industrial, los cuales corresponden a la industria automotriz y naval, respectivamente. En particular, el primer caso de estudio está basado en el problema de programación detallada de la producción de una empresa automotriz que procesa partes de automóviles, dicho problema fue abordado previamente por Roshanaei et al. (2013). Mientras que, el segundo ejemplo considera, de manera integrada, el problema de programación de operaciones de ensamblado y la reconfiguración de un astillero de una compañía naval dedicada a la construcción de grandes barcos. Si bien, en los capítulos anteriores de la presente tesis se ha abordado el mismo proceso productivo, en este capítulo se considera que el astillero se compone de estaciones o celdas de trabajos integradas por unidades multipropósito, logrando una mayor flexibilidad en el proceso productivo.

El algoritmo y los modelos matemáticos MILP se codifican en GAMS 24.9.2 utilizando CPLEX 12.7.1.0 como optimizador. Además, se utiliza como hardware una PC con procesador Intel Xeon X5650 de cuatro núcleos (2,6 GHz) y 32 GB de RAM.

### 4.5.1. Ejemplo ilustrativo

Se presenta un pequeño ejemplo académico para validar la estrategia de solución desarrollada. El proceso en cuestión involucra el procesamiento de $|I| = 9$ productos, de los cuales 3 son productos finales ($|I^f| = 3$). Específicamente, dichos productos finales se conforman por dos subensambles cada uno ($|I^a| = 6$): $SA_{i_7} = \{i_1, i_2\}$, $SA_{i_8} = \{i_3, i_4\}$, y $SA_{i_9} = \{i_5, i_6\}$. Como se muestra en la Figura 4.11, el proceso productivo consta de 4 estaciones de trabajo ($|U| = 4$), conformadas por un total de 6 unidades de procesamiento ($|K| = 6$), las cuales pueden realizar las operaciones de 3 etapas productivas ($|S| = 3$). Las unidades que realizan operaciones similares son agrupadas en estaciones de trabajo $u$. En este caso, existe una única unidad multipropósito $k_3$, correspondiente a la estación $u_2$, que puede realizar las operaciones de las etapas productivas $s_1$ y $s_3$.

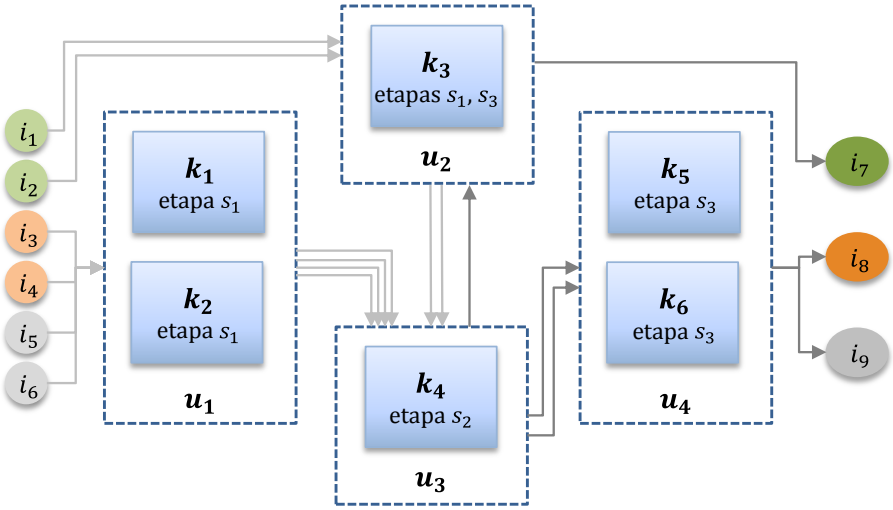



Figura 4.11. Esquema productivo del ejemplo ilustrativo.

En este ejemplo, el procesamiento de los productos $i_1 - i_6$ comienza en la estación de trabajo $u_1$ o $u_2$ con la primera etapa $s_1$, para luego, ser ensamblados de a pares en la estación $u_3$, donde se lleva a cabo la etapa $s_2$. Como resultado, se obtienen los productos

ensamblados $i_8$ e $i_9$, los cuales son finalmente procesados en la estación $u_4$, correspondiente a la etapa $s_3$, al igual que el producto $i_7$ que no requiere operaciones de montaje. Seguidamente, la Tabla 4.1 detalla la información relativa a los tiempos de procesamiento de todos productos en cada etapa de procesamiento.

Tabla 4.1. Tiempos de procesamiento (hora) – Ejemplo ilustrativo.

| Producto $i$ | Etapas productivas | | |
|---|---|---|---|
| | $s_1$ | $s_2$ | $s_3$ |
| $i_1$ | 4 | 9 | - |
| $i_2$ | 5 | 9 | - |
| $i_3$ | 5 | 4 | - |
| $i_4$ | 8 | 4 | - |
| $i_5$ | 3 | 7 | - |
| $i_6$ | 9 | 7 | - |
| $i_7$ | - | - | 10 |
| $i_8$ | - | - | 8 |
| $i_9$ | - | - | 6 |
| Unidades de procesamiento | $\boldsymbol{k_1, k_2, k_3}$ | $\boldsymbol{k_4}$ | $\boldsymbol{k_3, k_5, k_6}$ |

El problema planteado se resuelve, utilizando tanto la formulación matemática MILP, como con el algoritmo iterativo. Para este ejemplo, sólo se considera el problema de programación de las operaciones. Los resultados computacionales obtenidos se resumen en la Tabla 4.2. En dicha tabla, se puede observar que el algoritmo logra obtener una solución con un valor de la función objetivo de $MK = 31$ horas en 0,98 segundos de tiempo de CPU. Este valor coincide con la solución óptima encontrada por el modelo exacto MILP, después de 0,25 CPUs. A pesar de la mejor performance computacional reflejada por el enfoque exacto, este ejemplo resulta útil para mostrar la convergencia del algoritmo a la solución óptima del problema.

Tabla 4.2. Resultados computacionales - Ejemplo ilustrativo.

| Modelo MILP | | | | | | Algoritmo Iterativo | | |
|---|---|---|---|---|---|---|---|---|
| Var. Cont. | Var. Bin. | Ecs. | **Func. Obj.** | CPU ($s$) | Gap % | Solución inicial | **Mejor solución** | CPU Total (s) |
| 196 | 105 | 26 | **31** | 0,25 | 0 | 31 | **31** | 0,98 |

El programa óptimo reportado por ambos enfoques se muestra en la Figura 4.12 a través de un diagrama Gantt, ilustrando la asignación y secuenciación de las tareas en cada unidad de procesamiento. Notar que, cada tarea se identifica por el número de producto y el número de etapa, por ejemplo, la tarea "2-1" en la unidad $k_1$ hace referencia al procesamiento del producto $i_2$ en la etapa $s_1$. Además, el conjunto de tareas $t$ asociadas a un producto final $i$ ($t \in JT_i$) y a sus respectivos subensambles $i' \in SA_{i'}$, están representadas con el mismo color.

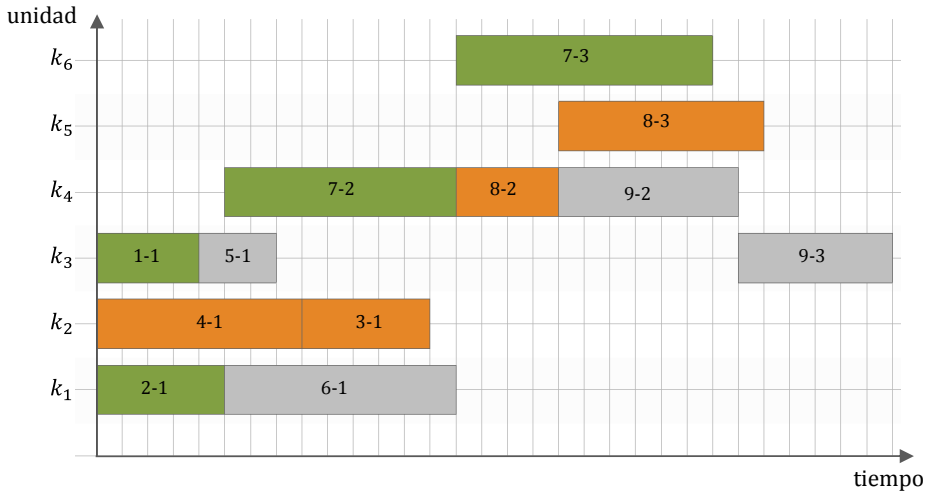



Figura 4.12. Programa de operaciones del ejemplo ilustrativo – $MK =$ 31 horas.

## 4.5.2. Caso de estudio 1: Industria Automotriz

El primer caso de estudio se basa en un ejemplo previamente abordado por Roshanaei et al. (2013), el cual comprende la producción y ensamblado de moldes para la industria automotriz. Según el proceso de manufactura, un molde está compuesto por 5

partes diferentes: (i) cavidad, (ii) estructura, (iii) corredera, (iv) retractor y (v) placas de sujeción. En la definición del problema, tanto los moldes como sus partes, representan productos que deben ser procesados a lo largo de una línea de manufactura. El caso de estudio original (Roshanaei et al., 2013) involucra el procesamiento de 20 partes, correspondientes a 4 moldes, en 8 etapas productivas con un total de 14 unidades de procesamiento, sin considerar la operación final de ensamblaje de moldes. En este capítulo, se amplía el estudio considerando una última etapa con 2 unidades en paralelo para el ensamblado final de dichos moldes.

La Tabla 4.3 muestra la secuencia de procesamiento para cada tipo de producto y las unidades que realizan las operaciones en cada etapa de procesamiento. De esta tabla, se deduce que un total de 96 tareas ($|T| = 96$) deben ser asignadas y secuenciadas para lograr un programa de operaciones factible. Además, se puede observar que el ejemplo cuenta con 14 unidades de procesamiento con propósito único y 2 unidades multipropósito ($k_3$ y $k_4$). La información relativa a los tiempos de procesamiento de los productos en cada etapa se especifica en la Tabla 4.4.

Tabla 4.3. Secuencia de procesamiento de cada producto – Caso de estudio 1.

| Tipo de Producto | Etapas de procesamiento | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
| *i. Cavidad* | x | x | x | x | x | x | x | x | |
| *ii. Estructura* | x | x | x | x | x | x | x | x | |
| *iii. Correderas* | x | x | x | x | | | | | |
| *iv. Retractor* | x | x | | | | | | | |
| *v. Placas de sujeción* | x | | | | | | | | |
| *vi. Molde* | | | | | | | | | x |
| Unidades de procesamiento | $k_1, k_2, k_3, k_4$ | $k_5$ | $k_3, k_4$ | $k_6, k_7$ | $k_8, k_9$ $k_{10}$ | $k_{11}$ | $k_3, k_{12}$ | $k_{13}, k_{14}$ | $k_{15}, k_{16}$ |

Para evaluar la escalabilidad del algoritmo propuesto, se analizan 3 instancias del caso de estudio 1: (i) caso original con la fabricación de 4 moldes, y luego, dos instancias

adicionales que consideran la fabricación de (ii) 6 moldes y (iii) 8 moldes, respectivamente. Se toma cómo hipótesis que, los nuevos moldes (de 5 a 8) son similares a los moldes presentados (de 1 a 4, respectivamente) en las Tablas 4.3 y 4.4.

Tabla 4.4. Tiempos de procesamiento (horas) – Caso de estudio 1.

| Producto $i$ | Etapas | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | $s_1$ | $s_2$ | $s_3$ | $s_4$ | $s_5$ | $s_6$ | $s_7$ | $s_8$ | $s_9$ |
| $i_1$- *cavidad* | 60 | 85 | 24 | 52 | 40 | 84 | 32 | 26 | |
| $i_2$- *estructura* | 79 | 80 | 24 | 70 | 84 | 84 | 47 | 60 | |
| $i_3$- *correderas* | 11 | 35 | 25 | 19 | | | | | |
| $i_4$- *retractor* | 16 | 40 | | | | | | | |
| $i_5$- *placas de sujeción* | 36 | | | | | | | | |
| $i_6$- *cavidad* | 62 | 85 | 48 | 52 | 45 | 84 | 21 | 26 | |
| $i_7$- *estructura* | 74 | 74 | 48 | 60 | 65 | 84 | 40 | 60 | |
| $i_8$- *correderas* | 30 | 20 | 7 | 20 | | | | | |
| $i_9$- *retractor* | 35 | 36 | | | | | | | |
| $i_{10}$- *placas de sujeción* | 45 | | | | | | | | |
| $i_{11}$- *cavidad* | 60 | 94 | 36 | 52 | 40 | 84 | 21 | 23 | |
| $i_{12}$- *estructura* | 72 | 74 | 90 | 60 | 59 | 84 | 46 | 54 | |
| $i_{13}$- *correderas* | 16 | 20 | 12 | 35 | | | | | |
| $i_{14}$- *retractor* | 14 | 16 | | | | | | | |
| $i_{15}$- *placas de sujeción* | 45 | | | | | | | | |
| $i_{16}$- *cavidad* | 55 | 76 | 36 | 54 | 54 | 36 | 32 | 26 | |
| $i_{17}$- *estructura* | 67 | 70 | 36 | 60 | 73 | 36 | 26 | 46 | |
| $i_{18}$- *correderas* | 23 | 21 | 20 | 39 | | | | | |
| $i_{19}$- *retractor* | 23 | 38 | | | | | | | |
| $i_{20}$- *placas de sujeción* | 50 | | | | | | | | |
| $i_{21}$- *molde* | | | | | | | | | 35 |
| $i_{22}$- *molde* | | | | | | | | | 30 |
| $i_{23}$- *molde* | | | | | | | | | 25 |
| $i_{24}$- *molde* | | | | | | | | | 38 |

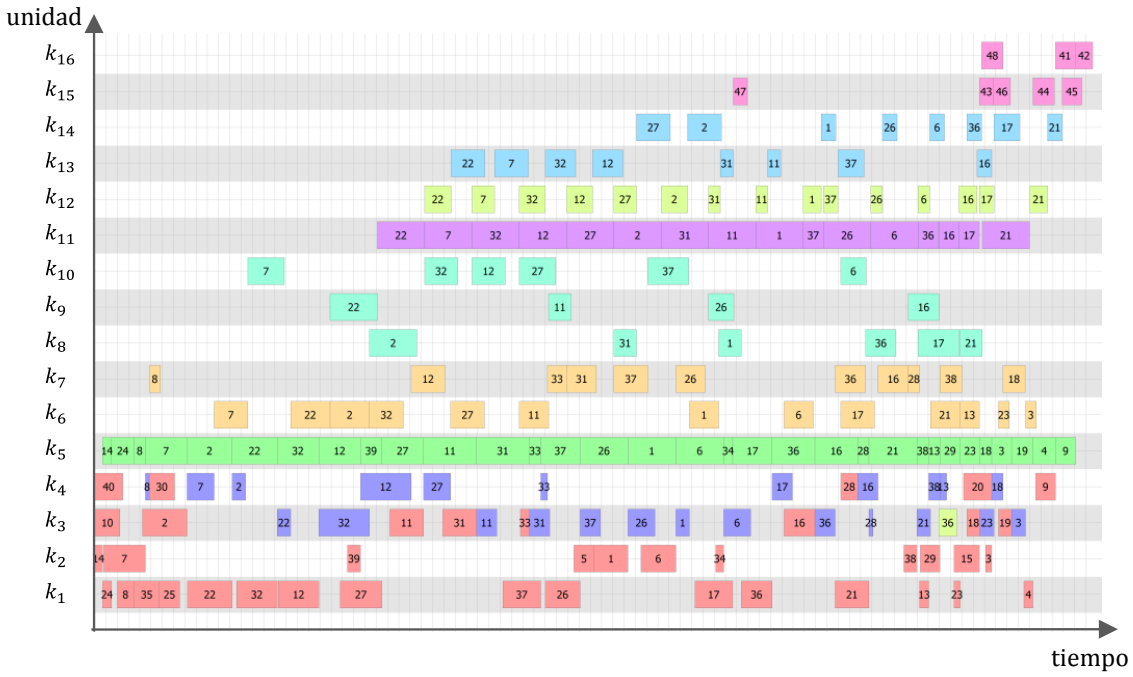
Las estadísticas computacionales y los valores de la función objetivo, para cada instancia y cada enfoque, se resumen en la Tabla 4.5. El enfoque exacto de optimización encuentra la solución óptima, para el problema de programación de "scheduling" original, en 127,7 segundos de tiempo de CPU, reportando un makespan de 979 horas. El algoritmo de descomposición reporta la misma solución para sólo 28,8 CPUs.

Tabla 4.5. Estadísticas computacionales – Caso de estudio 1.

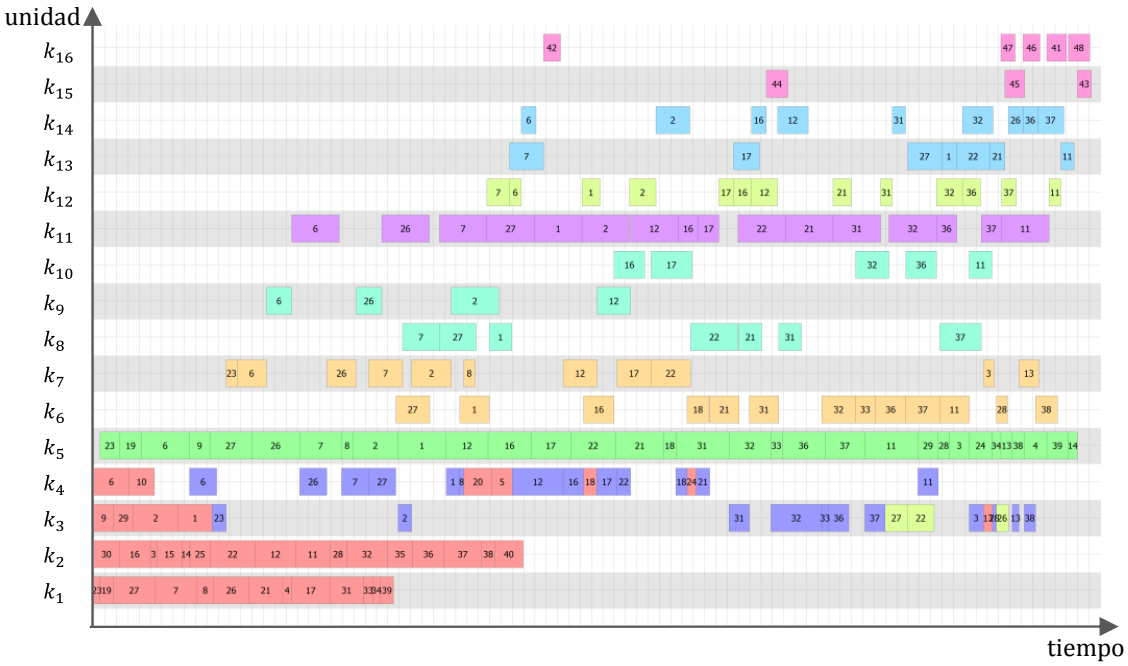
| Moldes | Modelo MILP | | | | | | Algoritmo Iterativo | | |
|---|---|---|---|---|---|---|---|---|---|
| | Var. Con. | Var. Bin. | Ecs. | **Func. Obj.** | CPU (*s*) | Gap % | Solución inicial | **Mejor solución** | CPU Total (s) |
| 4 | 194 | 2332 | 4685 | **979** | 127,7 | 0 | 1266 | **979** | 28,8 |
| 6 | 290 | 5166 | 10363 | **1355** | 3600 | 30,5 | 1633 | **1355** | 455,7 |
| 8 | 386 | 9112 | 18265 | **1772** | 3600 | 55,4 | 2030 | **1764** | 1145 |

Para el resto de las instancias del caso de estudio, se puede observar que, debido al aumento en el tamaño del modelo y por lo tanto, en la complejidad del problema, el enfoque monolítico no puede encontrar la solución óptima del problema dentro de un límite de tiempo de una hora de CPU. En particular, en el caso de 6 moldes, el modelo MILP reporta una solución de 1355 horas con un valor de GAP del 30,5%, mientras que el algoritmo alcanza la misma solución en sólo 455,7 segundos. En el caso de la producción de 8 moldes, el procedimiento de descomposición reporta el mejor "schedule" con un makespan de 1764 horas en 1145 CPUs, mientras que el enfoque exacto alcanza una solución de 1772 horas después de una hora de tiempo de CPU. Se debe notar que, encontrar una solución para el problema de 8 moldes implica la asignación y secuenciación de 192 tareas en 16 unidades de procesamiento. A pesar de la complejidad del problema, el enfoque de descomposición demuestra un buen rendimiento computacional, informando una mejor solución que la formulación MILP monolítica, con un tiempo de CPU aceptable.

En la Figura 13a y 13b se ilustran las mejores soluciones encontradas para la instancia de mayor tamaño (8 moldes), por cada estrategia de solución. Debido al espacio reducido disponible en estos diagramas Gantt, cada tarea se identifica sólo por número de producto $i \in I$ asociado. Por ejemplo, el molde 1 corresponde al producto $i_{48}$ y sus partes involucradas son $i_1, i_2, i_3\ i_4$ y $i_5$, que están representados en los diagramas Gantt de las Figuras 13a y 13b por #48, #1, #2, #3, #4 y #5, respectivamente. Además, las tareas asignadas a cada unidad de procesamiento se identifican con diferentes colores, según la etapa de procesamiento a la que correspondan. Por lo tanto, como se puede apreciar, las unidades multipropósito ($k_3$ y $k_4$) tienen tareas de diferentes colores. Tal es así que, en la mejor solución reportada por el enfoque iterativo, y representada en la Figura 13a, la unidad multipropósito $k_3$ procesa dos veces los productos $i_{11}$, $i_{31}$ y $i_{36}$ (#11, #31 y #36). Esto se debe a que dicha unidad realiza las operaciones de tres etapas de procesamiento: $s_1$, $s_3$ y $s_6$.




(a) $MK = 1764$ horas.




(b) $MK = 1772$ horas.

Figura 4.13. Programa de operaciones para 8 moldes – Caso de estudio 1: (a) Enfoque de descomposición y (b) Modelo MILP monolítico.

### 4.5.3. Caso de estudio 2: Industria Naval

El segundo caso de estudio se basa en un proceso de construcción de barcos que involucra varias etapas de procesamiento y unidades multipropósito. Como se ha detallado en los capítulos anteriores, dicho proceso se basa en el diseño modular de cada embarcación que se desea fabricar, por lo que, un barco se construye mediante el montaje de un conjunto específico de bloques y sub-bloques, donde ambos tipos de productos se consideran partes o productos intermedios.

Cabe destacar que, en los Capítulos 2 y 3 se ha estudiado el problema de "scheduling" para el proceso de construcción de barcos, considerado un sistema tipo *flow shop flexible*. En particular, en este capítulo se aborda la problemática de programación de operaciones y de rediseño de manera integrada, considerando el astillero como un sistema de manufactura tipo *job shop flexible*, es decir, que cuenta con estaciones de trabajo constituidas por unidades multipropósito. Al igual que el caso de estudio abordado en los

capítulos anteriores, se estudia el proceso de fabricación de un barco compuesto por 25 bloques (cada uno formado por 2 sub-bloques), que se lleva a cabo en un astillero que cuenta con 42 talleres (unidades de procesamiento) y 7 etapas de manufactura. En la configuración estudiada en este capítulo, se tienen en cuenta 13 unidades multipropósito:

- 6 unidades de procesamiento están destinadas a realizar las operaciones de dos etapas productivas: $s_2$ y $s_4$.
- 7 unidades pueden llevar a cabo las operaciones de tres etapas: $s_2$, $s_4$ y $s_6$.

La Figura 4.14 muestra la configuración descripta para el presente caso de estudio, considerando el procesamiento de 6 sub-bloques ($i_1 - i_6$) y 3 bloques ($i_7$, $i_8$ y $i_9$) a modo de ejemplo. Notar que, las unidades de procesamiento son agrupadas en 6 celdas de trabajo: algunas de propósito único ($u_1$, $u_3$, $u_5$ y $u_6$) y otras multipropósito, tales como $u_2$ (donde se llevan a cabo las etapas $s_2$ y $s_4$) y $u_4$ (correspondiente a las etapas productivas $s_2$, $s_4$ y $s_6$). Asimismo, se puede observar que, las celdas $u_1$ y $u_3$ sólo procesan sub-bloques y las celdas $u_5$ y $u_6$ sólo bloques, mientras que $u_2$ y $u_4$ pueden procesar ambos tipos de productos. Es importante recordar que, en la última etapa de procesamiento se debe cumplir un orden de montaje de bloques ($Seq_i$), específico para cada tipo de barco, y que en este caso coincide con el orden lexicográfico del conjunto $|I^f|$. Esta última condición, se ve reflejada en la secuencia de procesamiento de la etapa $s_7$ de la Figura 4.14: $i_7$-$i_8$-$i_9$.

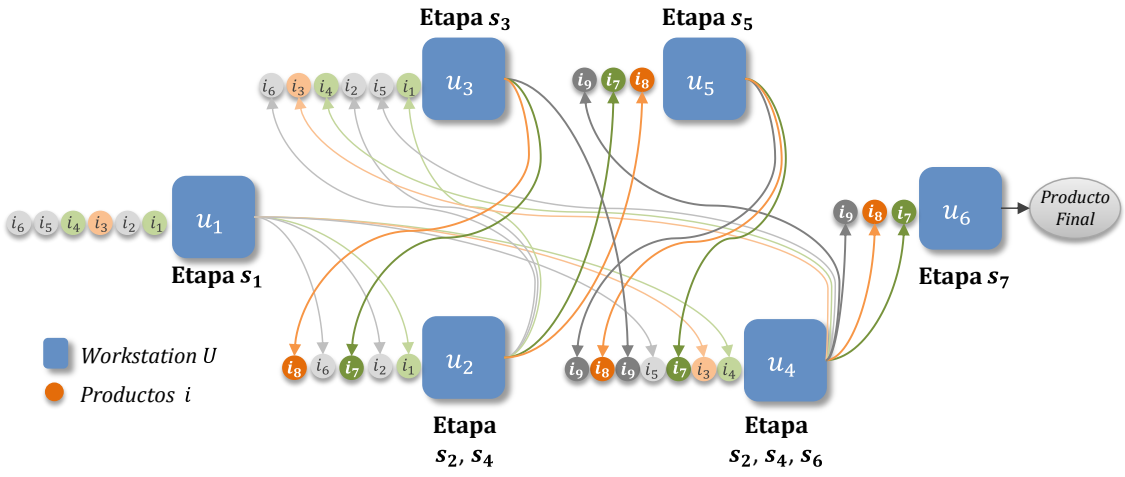



Figura 4.14. Diagrama de la configuración de la planta – Caso de estudio 2.

En la Tabla 4.6 se detallan los resultados computacionales obtenidos al resolver el problema de programación de operaciones, utilizando tanto el modelo MILP, como el algoritmo de descomposición. En particular, el modelo monolítico para el problema de "scheduling", está compuesto por 148551 restricciones, 9300 variables binarias y 452 variables continuas. Dicho modelo reporta un makespan de 241,7 días con un GAP del 25,4% después de cumplir con el límite de tiempo predefinido de una hora de tiempo de CPU. Por su parte, el algoritmo de descomposición permite mejorar dicha solución en un 2,8%, reportando un makespan de 235 días en tan sólo 265,2 segundos de tiempo de CPU.

Tabla 4.6. Estadísticas computacionales del problema de "scheduling" – Caso de estudio 2.

| Modelo MILP | | | Algoritmo Iterativo | | | |
|---|---|---|---|---|---|---|
| **Solución** | CPU (s) | GAP (%) | Solución inicial | CPU (s) | **Mejor solución** | CPU* (s) |
| **241,7** | 3600 | 25,2 | 261,2 | 18,9 | **235** | 265,2 |

* tiempo acumulado, que incluye etapas algorítmicas previas

Cabe destacar que, también se analiza el desempaño del modelo MILP monolítico definiendo como único criterio de terminación el reporte de la solución óptima, es decir, estableciendo un GAP del 0%. Como resultado, el "solver" termina al exceder la capacidad de memoria, prácticamente sin poder reducir el makespan en un día, ya que reporta un "schedule" con un makespan de 241,4 días y un valor de GAP del 25,3%.

La solución reportada por la formulación matemática MILP monolítica y la mejor solución encontrada para el problema de "scheduling" por el algoritmo de descomposición, se muestran en las Figuras 4.15 y 4.16, respectivamente, a través de diagramas Gantt. Notar que, un total de 225 tareas son asignadas y secuenciadas en 42 unidades de procesamiento, reflejando de esta manera la alta complejidad del ejemplo, y al mismo tiempo, la eficiencia del procedimiento de descomposición. Al igual que en los anteriores programas de operaciones, todas las tareas que pertenecen a una misma etapa de procesamiento se representan con el mismo color. De este modo, las unidades multipropósito tienen asignadas operaciones de diferentes colores.

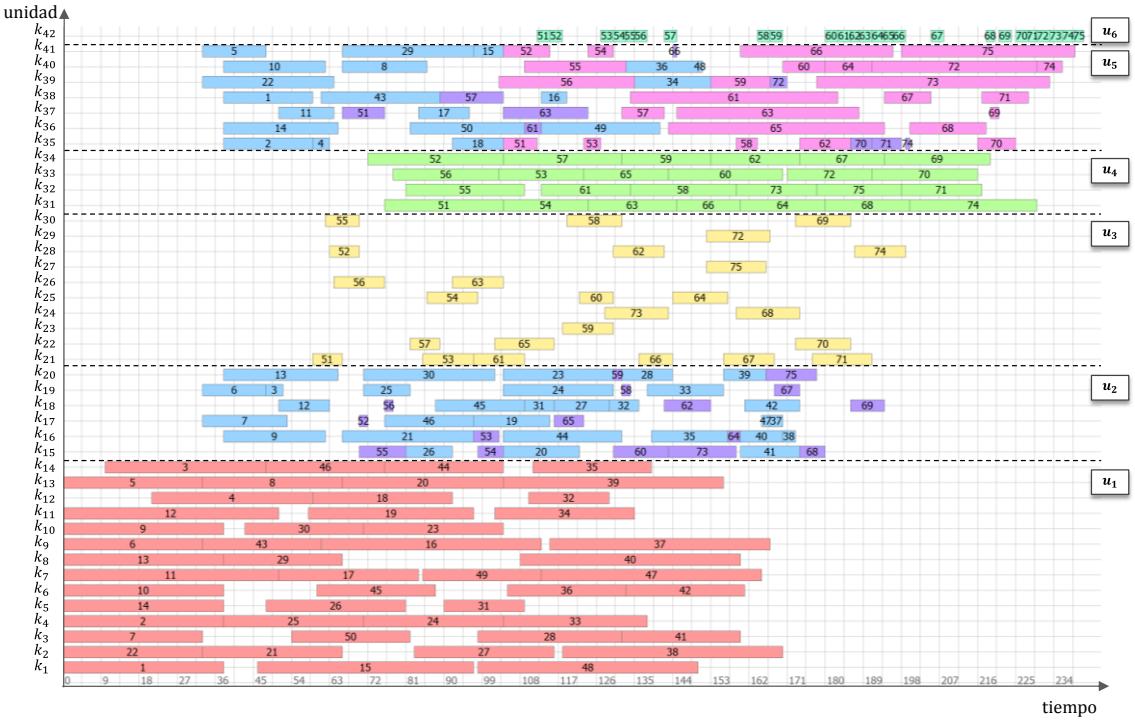



Figura 4.15. Solución reportada por el modelo MILP monolítico del caso de estudio 2 – $MK = 241{,}7$ días.

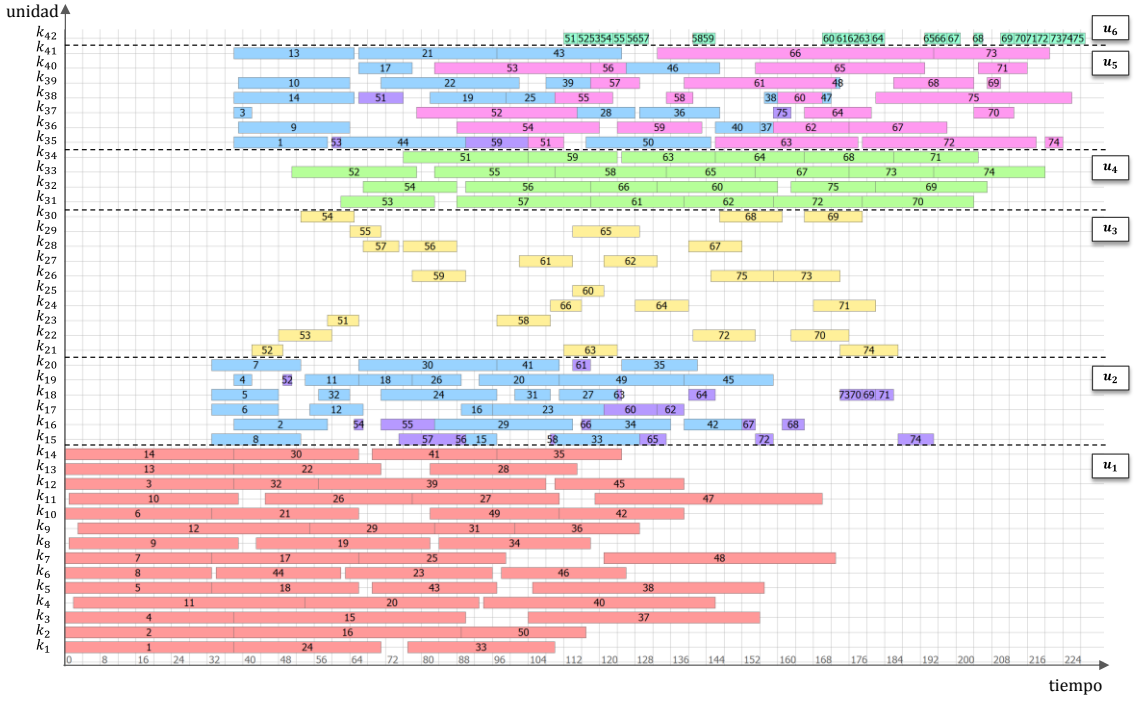



Figura 4.16. Mejor solución reportada por la etapa de programación del algoritmo de descomposición del caso de estudio 2 – $MK = 235$ días.

Por otra parte, en la Figura 4.17 se muestra la evolución de la solución reportada por ambos enfoques a lo largo del tiempo computacional. Se puede observar la alta eficiencia computacional que presenta la metodología iterativa desarrollada, la cual permite reportar en menos de 5 minutos de CPU una mejora en la solución del 84,7%, con respecto a la formulación MILP monolítica en el mismo tiempo de CPU. Particularmente, para el enfoque iterativo, se considera que los parámetros de la segunda sub-etapa de programación toman los siguientes valores: (i) $N^{max} = 10$ y (ii) $lTime = 10$ CPUs. De este modo, una vez más, se demuestra que la resolución del problema de "scheduling" de escala industrial, a partir de su descomposición en sub-problemas con espacios de búsqueda reducidos, permite encontrar una solución de buena calidad en un tiempo considerablemente menor que el método exacto monolítico, el cual tiene en cuenta el espacio de búsqueda completo.

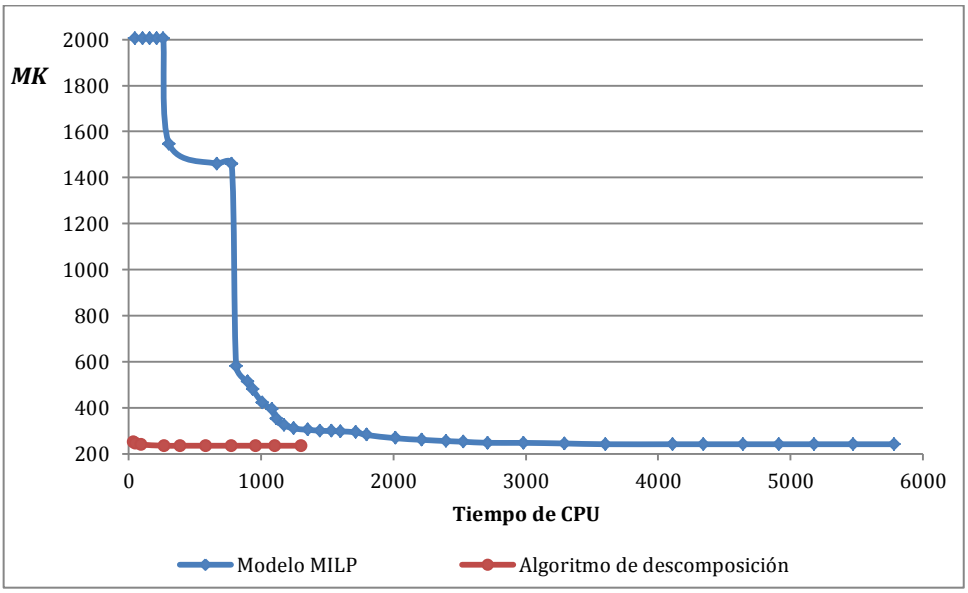



Figura 4.17. Evolución de las soluciones del Modelo MILP vs. Algoritmo de descomposición.

Para analizar el comportamiento del algoritmo de descomposición, ante un cambio en los parámetros de entrada de la fase de mejora: (i) $N^{max}$ y (ii) $lTime$, se realizan varias corridas del algoritmo iterativo. En la Figura 4.18 se muestran el comportamiento del algoritmo para el problema de "scheduling" para diversos valores de los parámetros

mencionados. Por un lado, se resuelve el problema considerando el "rescheduling" simultáneo de hasta 10 productos finales (bloques) y sus respectivos subensambles o productos intermedios (sub-bloques): $N^{max} = 10$. Por otro lado, se consideran 4 valores para el límite de tiempo de ejecución $lTime$: (i) 10, (ii) 20, (iii) 30 y (iv) 60 segundos de tiempo de CPU. En la Figura 4.18, se puede observar que, independientemente del valor que toma el parámetro $lTime$, en las primeras iteraciones el makespan se reduce considerablemente. Luego, a partir de un $N = 4$ ó $N = 5$, según corresponda, no se evidencian mejoras en el valor de la función objetivo.

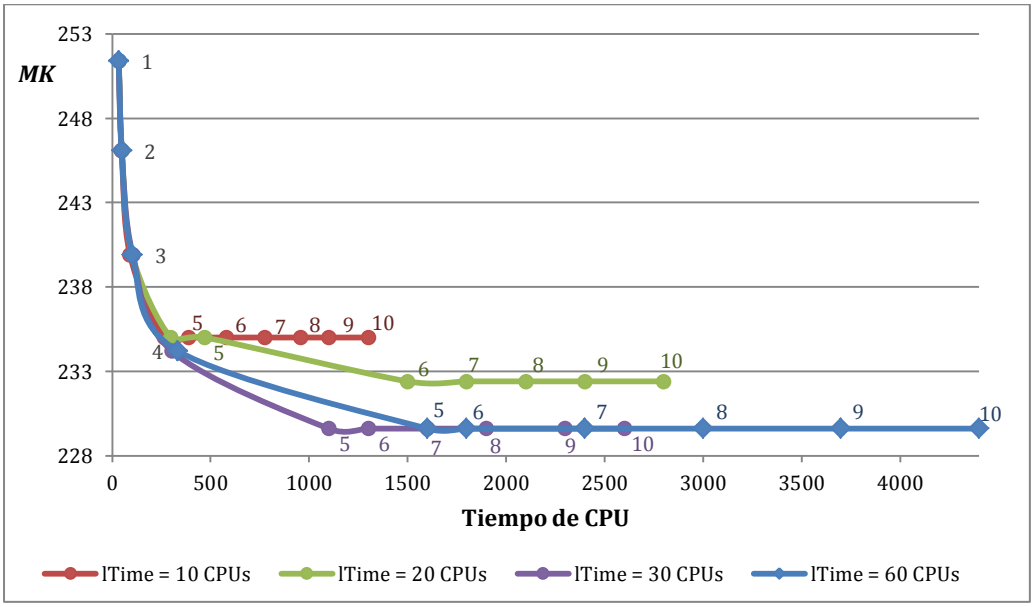



Figura 4.18. Calidad de la solución vs. Tiempo computacional – Algoritmo de descomposición.

Una vez obtenida la mejor solución para el problema de "scheduling", y observando que existen celdas de trabajo que se encuentran sobredimensionadas en cuanto a recursos, se procede a ejecutar la segunda etapa del procedimiento iterativo para analizar el rediseño del sistema productivo. Es importante mencionar que, a partir de los resultados expuestos en la Figura 4.18 para la etapa de programación, se toma el valor de los parámetros $N^{max} = 5$ y $lTime = 30$ segundos de CPU, buscando una buena calidad de la solución con un esfuerzo computacional razonable. Como resultado se obtiene, un makespan de 229,6 días en 1100 segundos de tiempo de CPU para la etapa de

programación de las operaciones, y la "liberación" de 7 unidades de procesamiento en la primera sub-etapa de rediseño: (i) $k_{16}$ y $k_{20}$ de la estación de trabajo $u_2$ y (ii) $k_{22}$, $k_{23}$, $k_{26}$, $k_{29}$ y $k_{30}$ de la estación $u_3$. En la Figura 4.19 se ilustra el programa de operaciones con la reducción de unidades, el cual evidencia la reasignación de tareas a otras unidades de procesamiento en paralelo. Comparando las soluciones mostradas en las Figuras 4.16 y 4.19, se puede inferir que la etapa de ensamblado $s_3$ se encuentra sobredimensionada debido a que la totalidad de las operaciones llevadas a cabo en la misma se pueden realizar en sólo 4 unidades, sin que conlleve un aumento del valor del makespan.

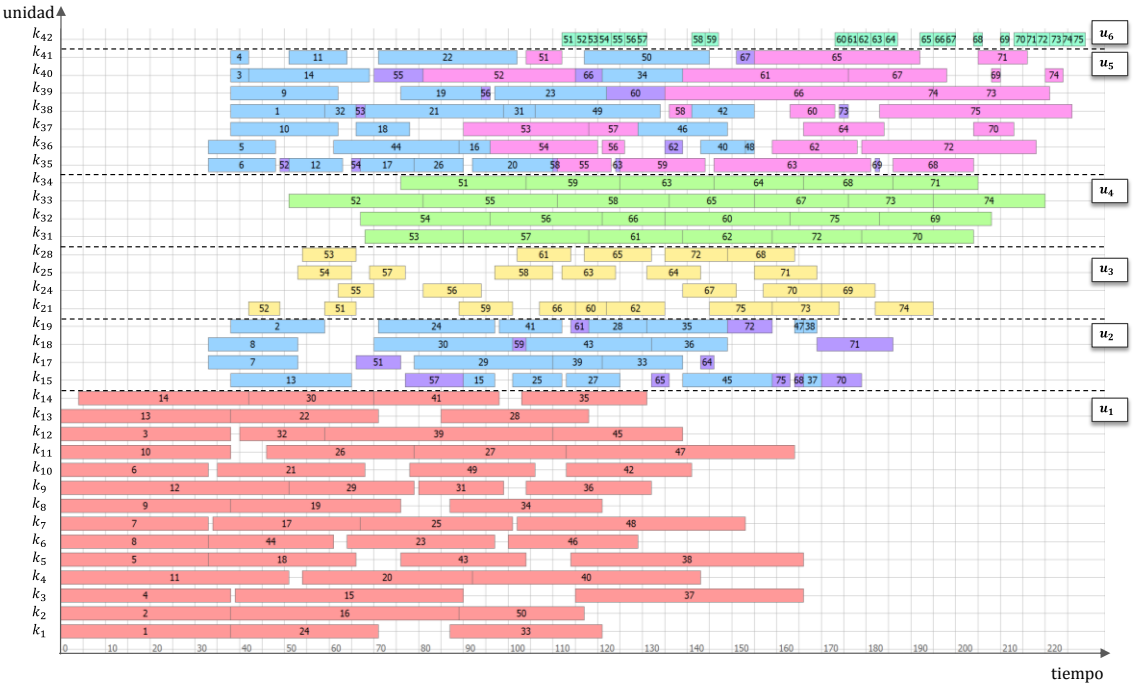



Figura 4.19. Mejor solución reportada por la primera sub-etapa de rediseño para el caso de estudio 2 – $MK = 229{,}6$ días.

Posteriormente, el algoritmo procede a la reubicación de las unidades de procesamiento liberadas $k \in L$. La Tabla 4.7 resume la información relativa a los resultados computacionales reportados aplicando las dos heurísticas propuestas. Asimismo, la Figura 4.20 muestra el diagrama Gantt correspondiente al "schedule" reportado por el algoritmo, aplicando la *heurística 1* para la fase de reubicación de unidades, donde las mismas son reasignadas a estaciones de trabajo una a una, de manera

iterativa. En este caso se obtiene un makespan de 217,5 días en menos de 4 horas de tiempo de CPU. Notar que, esta solución es muy similar a la reportada por el algoritmo al utilizar una reubicación simultánea de todas las unidades liberadas (*heurística 2*): 216,3 días, requiriendo menos de 5 horas de cómputo. Sin embargo, se puede obtener una reducción del makespan del 6,2% (13,4 días) al no considerar un límite de tiempo como criterio de terminación. Cabe destacar que, esta solución se obtiene después de aproximadamente 2 días de CPU, al terminar el "solver" debido al error de "capacidad de memoria excedida". En la Figura 4.21 se ilustra el "schedule" correspondiente a dicha solución. De este modo, el astillero puede aumentar su eficiencia productiva al reubicar 7 unidades de procesamiento y completar la construcción del barco requiriendo un total de 202,9 días.

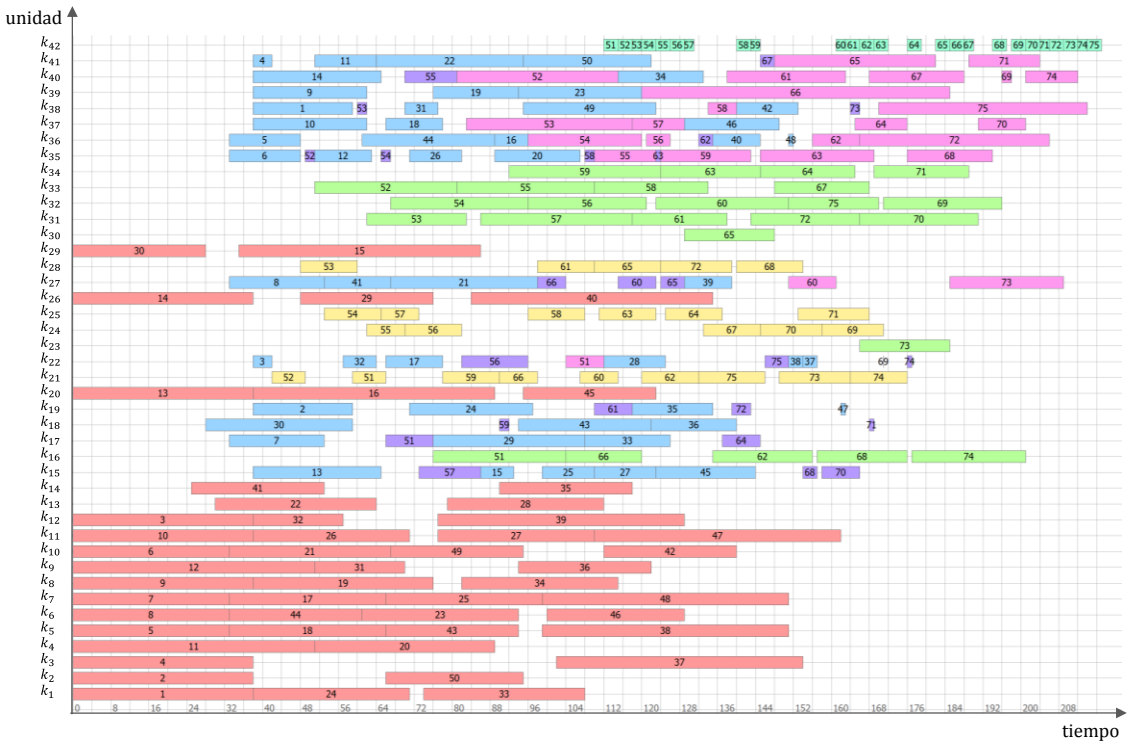



Figura 4.20. Mejor solución reportada por el algoritmo reasignando unidades y tareas para el caso de estudio 2 (*Heurística 1*) – $MK = 217{,}5$ días.

Tabla 4.7. Estadísticas computacionales del problema integrado de "scheduling" y rediseño – Caso de estudio 2.

| Heurística | Etapa de "scheduling" | | | | Etapa de rediseño | | | |
|---|---|---|---|---|---|---|---|---|
| | Solución inicial | CPU (s) | **Mejor solución** | CPU* (s) | Unidades liberadas | CPU* (s) | **Solución con rediseño** | CPU* (s) |
| 1 | 261,2 | 19,0 | 229,6 | 1139,3 | 7 | 12337,0 | 217,5 | 12497,7 |
| 2 | 261,2 | 18,1 | 229,6 | 1147,0 | 7 | 12355,7 | 216,3 | 15957,7 |
| 2** | 261,2 | 17,8 | 229,6 | 1141,7 | 7 | 12353,8 | 202,9 | 188967,6 |

* tiempo acumulado, que incluye etapas algorítmicas previas

** sin límite de tiempo de ejecución de la última fase de la etapa de rediseño, termina el "solver" por exceder la capacidad de memoria

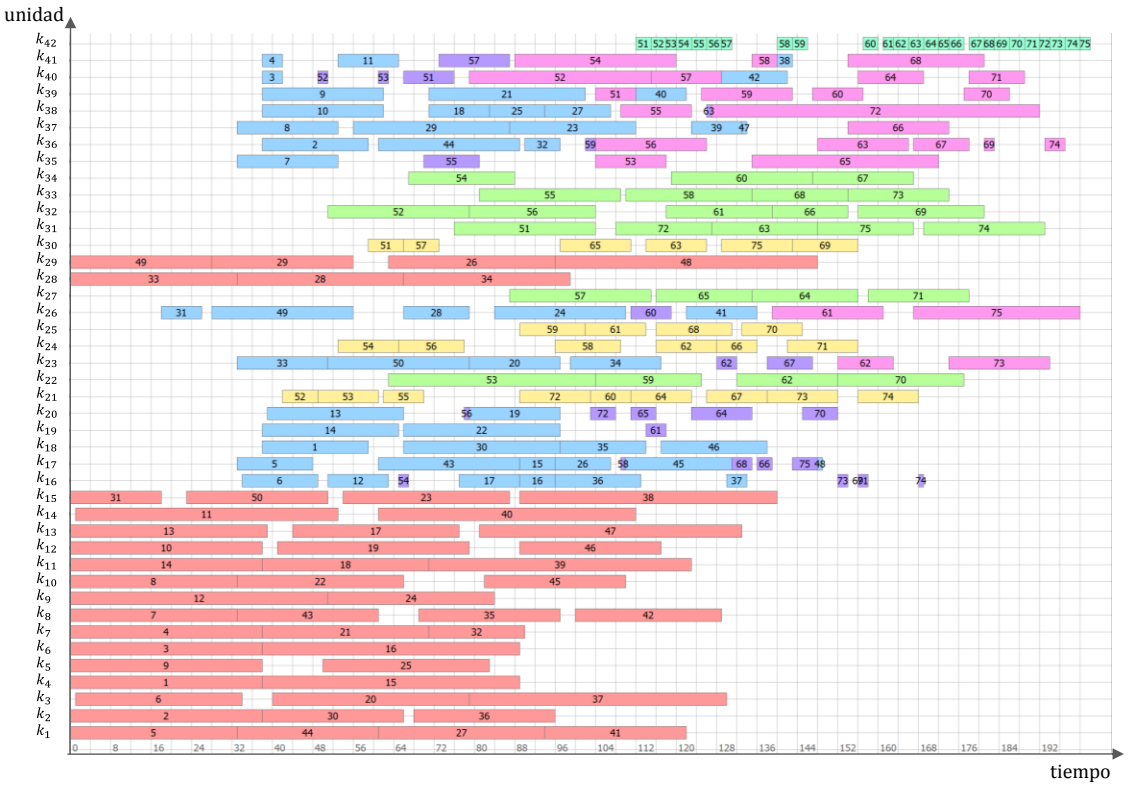



Figura 4.21. Mejor solución reportada por el algoritmo reasignando unidades y tareas para el caso de estudio 2 (*Heurística 2*) – $MK = 202{,}9$ días.

## 4.6. Conclusiones

En este capítulo se ha abordado, de manera integrada, el problema de programación de operaciones y rediseño de plantas de manufactura flexibles con unidades multipropósito, más conocidas como plantas job shop flexibles, donde además se consideran operaciones de ensamblado. Primeramente, se desarrolló un modelo matemático MILP riguroso monolítico, basado en el concepto de precedencia general global, para resolver la problemática de "scheduling" planteada. El modelo incorpora la definición de "tareas" para manejar las decisiones de asignación y de secuenciación de operaciones en unidades multipropósito, y restricciones adicionales relacionadas al rediseño de la planta.

Se estableció como objetivo la minimización del makespan y la reconfiguración de la planta, a fin de aumentar la eficiencia en la utilización de las unidades de procesamiento utilizadas. Razón por la cual, se establece como principal objetivo la minimización del tiempo total requerido para programar todos los productos involucrados en el problema bajo estudio, para luego minimizar las unidades necesarias para satisfacerlo y evaluar la reubicación de las mismas a otras celdas de trabajo compatibles, permitiendo un mejor uso de los recursos y una mejora en el makespan.

Como se ha demostrado en los anteriores capítulos, en los problemas resueltos, utilizando formulaciones matemáticas monolíticas, tanto el esfuerzo computacional como la calidad de la solución encontrada, dependen del tamaño del modelo, es decir, de la cantidad de productos que se deben asignar y secuenciar, las etapas y las unidades de procesamiento involucradas en la configuración del problema. Por lo tanto, debido a que los procesos productivos de escala industrial, particularmente de manufactura flexibles, implican el procesamiento de una gran variedad de productos, cada uno de los cuales puede seguir diferentes secuencias de procesamiento a través de la planta, y que en ciertos casos requieren operaciones de ensamblado para obtener el producto final, presentan una alta complejidad combinatoria. Por esta razón, se extiende el uso del modelo MILP desarrollado a aplicaciones del mundo real a través una metodología algorítmica de

descomposición, que resuelve de manera iterativa el modelo de optimización propuesto pero considerando un espacio de búsqueda reducido en cada ejecución del "solver".

En una primera etapa de la estrategia de solución propuesta, se genera un programa de operaciones inicial factible. Luego, este "schedule" es mejorado de manera iterativa a partir de reprogramaciones de subconjuntos de tareas, es decir, se toman decisiones de reasignación y reordenamiento de un subconjunto de tareas de manera simultánea en cada iteración, con el objetivo de minimizar el makespan. Cabe señalar que, la cantidad de tareas que se reprograman se incrementa en cada iteración, con el fin de ofrecer una mayor flexibilidad en las decisiones de "rescheduling". De esta manera, una solución que fue hallada aplicando un cierto criterio de pre-ordenamiento o después de un cierto tiempo de cómputo, puede ser gradualmente mejorada de manera iterativa, requiriendo un tiempo de cómputo aceptable. El bajo costo computacional que tiene esta estrategia se debe principalmente a que gran parte de las decisiones tomadas en cada iteración deben ser mantenidas durante la siguiente iteración. En otras palabras, esta estrategia permite sólo la realización de cambios parciales en el programa de operaciones utilizado como solución inicial en cada iteración.

Posteriormente, en una segunda fase, el algoritmo considera las decisiones de rediseño sobre la solución dada en la etapa anterior, de manera de aumentar la eficiencia del uso de las unidades de procesamiento, y al mismo tiempo, mejorar la solución del problema de "scheduling". De este modo, permite determinar si la planta se encuentra sobredimensionada al minimizar el número de unidades que se utilizan en cada estación de trabajo y analizar, si es posible, la reasignación de dichas unidades a otras estaciones de trabajo. Para este último paso, se proponen dos heurísticas para la reubicación de las unidades "liberadas": (i) reubicación iterativa de unidades de procesamiento y (ii) reubicación simultánea de todas las unidades liberadas.

Por último, el enfoque propuesto, se aplicó en instancias de diferentes tamaños derivadas de dos casos de estudio presentes en la industria automotriz y naval, respectivamente. Para determinar la complejidad de los ejemplos y la calidad de las

soluciones reportadas por el método de descomposición, también se ha aplicado como método resolutivo el modelo original MILP monolítico. En consecuencia, se pudo demostrar que el algoritmo converge eficientemente a la solución óptima con un modesto esfuerzo computacional cuando se consideran instancias de tamaño pequeño a mediano. Además, el procedimiento iterativo alcanzó mejores soluciones que el modelo matemático riguroso para problemas de gran escala, y al mismo tiempo, reflejó una alta eficiencia computacional. Por lo tanto, se puede concluir que el mismo resulta altamente competitivo frente a otras herramientas computacionales, no sólo por la calidad de las soluciones sino también por el esfuerzo computacional requerido para alcanzarlas.

## *Nomenclatura*

### *Conjuntos*

| | |
|---|---|
| $I$ | conjunto de órdenes de producto (índice $i, i'$) |
| $I^f$ | conjunto de productos finales |
| $I^a$ | conjunto de productos intermedios o subensambles |
| $SA_i$ | conjunto de subensambles del producto final $i$ |
| $S$ | conjunto de etapas de procesamiento (índice $s, s'$) |
| $S_i$ | conjunto de etapas donde se procesa el producto $i$ |
| $S^a$ | conjunto de etapas de ensamblado |
| $S^a_{seq}$ | conjunto de etapas de ensamblado que deben seguir un orden de montaje |
| $K$ | conjunto de equipos o unidades de procesamiento (índice $k$) |
| $K_s$ | conjunto de unidades disponibles para realizar las operaciones de la etapa de procesamiento $s$ |

| | |
|---|---|
| $L$ | conjunto de unidades "liberadas" (índice $k$) |
| $U$ | conjunto de estaciones o celdas de trabajos (índice $u$) |
| $U_k$ | conjunto de estaciones de trabajos compatibles con la unidad de procesamiento $k$ |
| $K_u$ | conjunto de unidades de procesamiento que forman parte de la estación de trabajo $u$ |
| $S_u$ | conjunto de etapas productivas que se llevan a cabo en la estación de trabajo $u$ |
| $T$ | conjunto de tareas (índice $t$) |
| $JT_i$ | conjunto de tareas asociadas al producto $i$ |
| $ST_s$ | conjunto de tareas asociadas a la etapa de procesamiento $s$ |

***Parámetros***

| | |
|---|---|
| $tp_{is}$ | tiempo de procesamiento del producto $i$ en la etapa $s$ |
| $tp_t$ | tiempo de procesamiento de la tarea $t$ |
| $Seq_i$ | secuencia de los productos $i$ |
| $M$ | constante big-M |
| $N$ | cantidad de productos finales a ser reprogramados simultáneamente en cada iteración |
| $N^{max}$ | cantidad máxima de productos finales a ser reprogramados simultáneamente en una iteración |
| $iter$ | número de iteración en la etapa algorítmica de programación de las operaciones |

$active_t$ toma valor verdadero si la tarea $t$ debe ser programada o reprogramada en una iteración

$active_k$ toma valor verdadero si la unidad $k$ debe ser reubicada en una estación de trabajo

$iterU$ número de iteración que identifica la estación de trabajo a ser evaluada en una iteración

$iterUnit$ número de iteración que identifica la unidad de procesamiento a ser reubicada en una iteración

$sMK$ último valor reportado del makespan

$sY_{tk}$ decisiones de asignación de la tarea $t$ en la unidad de procesamiento $k$

$sW_{tt'}$ decisiones de secuenciación de las tareas $t$ e $t'$ en la unidad de procesamiento $k$

$sTf_t$ decisiones de tiempo de finalización de la tarea $t$

$BestSol$ mejor solución reportada, correspondiente al valor del makespan

$lTime$ tiempo límite de CPU

***Variables Continuas***

$Ts_t$ tiempo de inicio de procesamiento de la tarea $t$

$Tf_t$ Tiempo de finalización de la tarea $t$

$V_k$ toma un valor mayor que cero cuando se utiliza la unidad de procesamiento $k$

$X_{ks}$ toma un valor mayor a cero cuando se selecciona la unidad de procesamiento $k$ para procesar la etapa $s$

$X_{ku}$ toma un valor mayor a cero cuando se la unidad de procesamiento $k$ es reubicada en la estación de trabajo $u$

$MK$ makespan

***Variables binarias***

$W_{tt'}$ toma valor 1 si la tarea $t$ es procesada antes que la tarea $t'$

$Y_{tk}$ toma valor 1 si la tarea $t$ es procesada en la unidad de procesamiento $k$

# Capítulo 5

# *Programación detallada de operaciones de procesos industriales con consumo intensivo de energía eléctrica*

## 5.1. Introducción

Actualmente, la competitividad de las industrias con procesos continuos y consumo intensivo de energía eléctrica (CIEE) está muy ligada a la capacidad que estas tienen para ajustar su producción de acuerdo a los precios de la electricidad, los cuales tienden a ser muy cambiantes a lo largo del día. En este contexto, la gestión dinámica de la demanda de electricidad, también conocida como "*Demand Side Management*" (DSM), surge como un enfoque eficaz para mejorar el rendimiento de la red eléctrica y aumentar los beneficios para el consumidor. Para lograr una gestión industrial de la demanda es necesario integrar eficientemente la producción y la administración de la energía eléctrica, lo que requiere una comprensión detallada del proceso de producción, así como un conocimiento sobre la economía del sistema de energía. Por lo tanto, para explotar las

oportunidades de DSM, resulta esencial una gestión eficiente de los complejos mecanismos que forman parte de los mercados de electricidad desregulados, que son diferentes a los mercados de productos básicos típicos en las industrias de proceso. Particularmente, en industrias con un alto consumo de electricidad, DSM se vuelve crítico al momento de mantener la rentabilidad, como afirman en su trabajo Zhang y Grossmann (2016a).

En particular, las grandes unidades de separación de aire ("Air Separation Units", ASU por sus siglas en inglés) son un ejemplo típico de procesos intensivos en energía (CIEE), donde se necesitan compresores de aire de potencia eléctrica de grandes dimensiones para alcanzar las temperaturas criogénicas necesarias en el proceso de obtención de oxígeno, nitrógeno, argón y otros gases. En la actualidad, las *Plantas de Separación de Aire* (PSA) se utilizan para la producción de dichos gases con grados de pureza muy elevados: nitrógeno hasta el 99,99 %, oxígeno hasta el 99,5 % y argón con pureza de hasta el 99,99 %, para distintas áreas de la industria, tales como la industria metalúrgica, electrónica, energética, química, petroquímica, de procesamiento del petróleo y fabricación de vidrios, así como para las empresas del área de venta de gases técnicos.

Debido a la volatilidad existente en los mercados de energía, las compañías tienen la gran oportunidad de beneficiarse en los períodos de precios más bajos de electricidad, reduciendo los costos asociados. En los últimos años se ha desarrollado una gran cantidad de modelos de optimización para enfrentar el desafío respecto a los problemas de programación que surgen en la comunidad PSE ("Process Systems Engineering"). Se pueden encontrar revisiones detalladas de esta área en el trabajo de Méndez et al. (2006), y más recientemente en el de Harjunkoski et al. (2014).

Particularmente, en el área de optimización de la energía a través de la programación detallada de las operaciones de producción, existen varias contribuciones importantes en la última década. Inicialmente, Castro et al. (2009) propone una formulación de programación de tiempo continuo para plantas continuas que operan bajo

un costo de electricidad variable. A su vez, Mitra et al. (2012) presentan un modelo MILP para la planificación de la producción óptima para procesos continuos con consumo intensivo de energía. Dicha formulación permite un modelado preciso y eficiente de las transiciones entre los modos de operación de una planta, usando una representación de tiempo discreto. Además, en dicho trabajo, el modelo se aplica con éxito a dos plantas industriales diferentes, una dedicada a la separación de aire para suministrar productos en estado líquido al mercado, y otra, dedicada a la producción de cemento. Un año después, Mitra et al. (2013) desarrollan un modelo de optimización generalizado de modos de operaciones, sobre la base de un componente, para la programación óptima de operaciones de plantas que combinan calor y energía bajo precios de electricidad sensibles a los períodos de tiempo. Este modelo es capaz de rastrear los estados de los componentes en términos de modos de operación y comportamientos de transición. Los autores lo aplican y evalúan para el caso de una planta de cogeneración industrial, obteniendo hasta un 20 % de aumento de ganancia en comparación con una operación constante de la planta.

Por otra parte, Artigues et al. (2013) proponen un enfoque de programación de restricciones de dos etapas, con el fin de resolver un caso de estudio real que involucra restricciones energéticas y objetivos relacionados con el consumo de energía eléctrica. Posteriormente, Mitra et al. (2014a) introducen un modelo de múltiple escala para la optimización integrada de inversiones y operaciones de procesos continuos CIEE, teniendo en cuenta precios de electricidad sensibles al tiempo e incertidumbre en la demanda. La formulación desarrollada es aplicada para una planta de separación de aire (PSA), considerando una demanda determinística y por otro lado, una demanda estocástica. Debido a la naturaleza del problema (escala múltiple), las formulaciones MILP resultantes tienden a ser muy grandes y difíciles de resolver. Para contrarrestar esta situación, los autores delinean un algoritmo híbrido de descomposición de dos niveles, que se presenta en la segunda parte de su trabajo (Mitra et al., 2014b), el cual es capaz de reducir el tiempo computacional en hasta dos órdenes de magnitud en comparación con el

método exacto, y en un 45-85 % en comparación con el enfoque de descomposición Benders. Vale la pena mencionar que, la descomposición de Benders es una técnica de programación matemática utilizada para resolver ciertos problemas de optimización a gran escala, debido a que el problema se divide en múltiples problemas más pequeños y de este modo, ofrece mejoras significativas en el rendimiento computacional.

Otras contribuciones también se focalizan en los casos de plantas PSA que operan en diferentes mercados de energía, como el caso de Pattison et al. (2016), donde los autores proponen un modelo dinámico para abordar el funcionamiento de procesos industriales en aquellos mercados de energía donde los precios varían en función del tiempo. Por otro lado, Zhang et al. (2016a) presentan un modelo general de tiempo discreto para la programación de operaciones de acuerdo a diversos contratos de energía. El modelo propuesto consiste en una formulación de programación basada en una red de procesos. En este sentido, se considera que un modelo de contrato de bloque está constituido por una gran variedad de contratos de energía de uso común. Posteriormente, los mismos autores (Q. Zhang et al., 2016b) se enfrentan a la optimización simultánea de la programación de producción a corto plazo y a la adquisición de electricidad en condiciones de incertidumbre, para procesos continuos CIEE. Al mismo tiempo, Zamarripa et al. (2016) desarrollan dos enfoques para abordar, de manera simultánea, dos temáticas: la producción y la distribución de las cadenas de suministro de gases industriales, teniendo como hipótesis que los precios de energía se encuentran regulados. Estos enfoques permiten determinar la programación agregada para horizontes rodantes en el tiempo. Recientemente, se ha informado a modo de resumen, en el trabajo realizado por Zhang y Grossmann (2016), los trabajos existentes sobre planificación y programación para el DSM industrial, junto con los principales mecanismos de los mercados eléctricos.

En este contexto, a pesar de que se han logrado avances muy significativos en el área de optimización de la programación de la producción operacional en procesos continuos CIEE, sigue siendo un problema contar con una estrategia de solución altamente eficiente y sistemática para resolver problemas industriales de “scheduling” a gran escala.

Motivado por las actuales limitaciones previamente descriptas, en este capítulo se presenta una formulación matemática MILP, con representación discreta del tiempo, conceptualmente diferente a las ya existentes en la literatura. El principal objetivo es establecer el plan detallado de operaciones de manera predictiva, es decir, determinar los modos de operación y los niveles de producción de una planta PSA por hora, al mismo tiempo que se satisface la demanda y se minimiza el costo energético total, para un horizonte de planificación determinado.

El modelo matemático desarrollado es capaz de tratar eficientemente las fluctuaciones de los precios del mercado sensibles a los períodos de tiempo, mientras que se optimizan las decisiones operativas correspondientes a los procesos de consumo intensivo de energía, incurriendo en un tiempo computacional modesto. El problema a resolver involucra, generalmente, previsiones de precio de electricidad, demanda y diversas restricciones operativas, tales como modos o estados de transición factibles, tiempos de residencia mínimo y máximo, niveles de inventario permitidos y tasas de producción mínimas y máximas, basadas en el estado de planta.

La estructura de este capítulo está organizada de la siguiente manera. En la sección 5.2 se describe de manera detallada el problema de programación de procesos CIEE, donde se incluye una descripción del funcionamiento de una planta PSA y los mercados de energía existentes. Luego, en la sección 5.3 se introduce una nueva representación para el proceso de transición de estados de operación del tipo de planta bajo estudio, llamado "*Process State Transition Network*" (PSTN). El modelo MILP basado en este esquema de transiciones se introduce en la sección 5.4. Además, se presenta un modelo MILP alternativo de la literatura, con el propósito de evaluar el rendimiento computacional de la formulación matemática desarrollada. La aplicabilidad y efectividad de estas formulaciones son comparadas e ilustradas a través de la resolución de problemas reales, presentados en la sección 5.5, donde también se reportan los principales resultados obtenidos para cada caso. La sección 5.6 resume las conclusiones y las mayores contribuciones realizadas. Por último, se detalla la nomenclatura utilizada.

## 5.2. Descripción del problema

El problema de programación de operaciones abordado en este capítulo se focaliza en los procesos con consumo de energía de manera intensiva. Generalmente, estos procesos involucran un conjunto de modos o estados específicos de producción correspondientes a la operación de grandes equipos, donde existen fuertes interacciones energéticas. De este modo, cambios relativamente pequeños en sus variables de decisión pueden dar lugar a oportunidades importantes de ahorro de energía eléctrica, y por lo tanto, en los costos asociados.

A continuación, en la sección 5.2.1, se describe el proceso productivo que se lleva a cabo en una planta de separación de aire, donde se utilizan grandes cantidades de energía eléctrica para la producción de diferentes productos, tales como oxígeno, nitrógeno líquido, etc. La programación de la producción de este tipo de plantas se ajusta a los esquemas de fijación de precios de electricidad, los cuales son altamente dependientes de los períodos de tiempo. Por esta razón, un factor clave que influye directamente en las decisiones operativas, es la volatilidad del precio de la energía (Karwan y Keblis, 2007; Mitra et al., 2013). Se debe tener en cuenta que, los usuarios pueden comprar energía en la red eléctrica mediante el uso de contratos alternativos. Los diferentes tipos de mercados existentes y cómo pueden diferir en cuanto al precio y la disponibilidad, se describen a continuación, en la sección 5.2.2.

### 5.2.1. Procesos industriales de consumo intensivo de energía

Los procesos continuos de separación de aire de alta potencia utilizan como principales suministros el aire de la atmósfera y la energía eléctrica. El aire es una materia prima abundante que se obtiene sin costo alguno, la cual se comprime y se seca para llevar a cabo una separación criogénica con el fin de obtener gases industriales, de acuerdo con las especificaciones de calidad requeridas, según corresponda.

La composición del aire seco es predominantemente 78 % de nitrógeno, 21 % de oxígeno y el 1 % restante está constituido por diferentes gases nobles, donde se destaca el argón. Para comprimir el aire, desde la presión atmosférica hasta presiones más elevadas, se utiliza un compresor principal. Dicho compresor puede ser complementado con compresión adicional seguida de expansión, proporcionando refrigeración adicional y de esta manera, producir productos líquidos y/o de alta presión. Después de la compresión, el aire se enfría y se licua parcialmente en un intercambiador de calor. Este aire, parcialmente condensado, ingresa a una columna de destilación criogénica, donde se separa en sus componentes básicos. Se debe tener en cuenta que, en ciertas plantas, la corriente de nitrógeno gaseoso puede enviarse a un ciclo de licuefacción, donde el gas se enfría hasta alcanzar una fase líquida. Generalmente, los licuefactores están formados por un compresor de alimentación, un compresor de reciclaje, un amplificador de turbina de expansión, un intercambiador de calor y un separador.

La elevada potencia utilizada para manejar el equipo de compresión, destinado a la producción de gases y líquidos (nitrógeno, oxígeno y argón), representa el mayor costo de operación individual en el proceso de separación de aire. Por esta razón, este capítulo se centra principalmente en el funcionamiento de una planta PSA, particularmente, en hallar los niveles óptimos de consumo de energía eléctrica de la misma, considerando el rol clave que tiene el proceso de compresión. De este modo, se busca resolver el problema de la programación de la producción, haciendo énfasis en los diferentes modos de operación de los compresores involucrados en la alimentación y reciclaje, y en las posibles transiciones que pueden ocurrir entre los mismos.

La Figura 5.1 muestra una representación esquemática de las configuraciones factibles y las transiciones permitidas en una planta típica dedicada a la separación de aire. Como se puede observar, dicha planta debe atravesar una serie de estados, tanto para la puesta en marcha, como para el apagado de los equipos. Asimismo, cuenta con estados de operación que requieren una permanencia de los equipos, sin cambio alguno, durante un tiempo mínimo. Dichos estados, y sus correspondientes duraciones, dependen de la

configuración de cada planta, para el caso del ejemplo de la figura se definen 3 estados, cada uno con una duración mínima de 3 horas: (i) modo de operación "*ON*" (funcionamiento normal de la planta), (ii) modo "*En espera*" (tiempo de espera, aunque los equipos no estén apagados, no hay producción) y por último, (iii) modo "*OFF*" (tiempo de inactividad, donde todos los equipos de la planta se encuentran apagados).

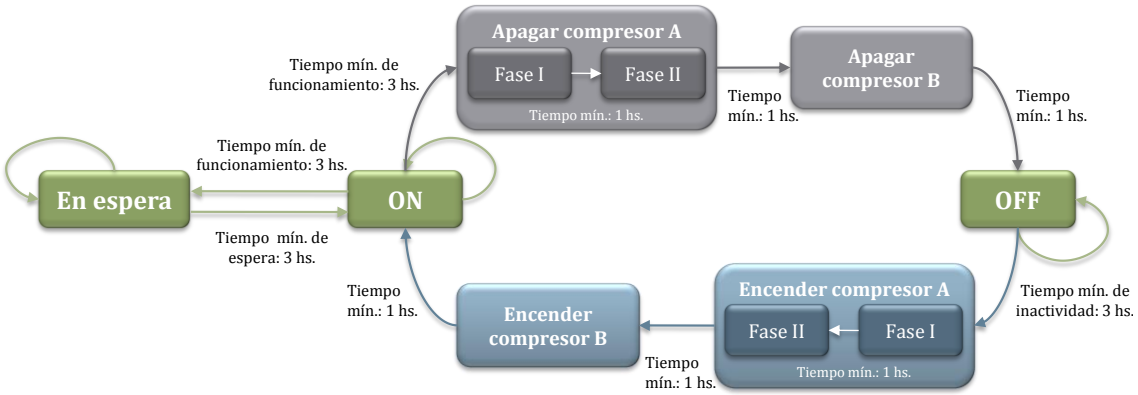



Figura 5.1. Esquema de estados de operación de la planta PSA.

A partir de la Figura 5.1, se pueden apreciar cinco modos de funcionamiento posibles en una planta PSA, los mismos se describen con mayor detalle a continuación:

- **Procedimiento de puesta en marcha.** Se utilizan tres modos de operación para modelar el arranque del equipamiento de la planta: "*Encender compresor A - Fase I*", "*Encender compresor A - Fase II*" y "*Encender compresor B*". Los equipos deben cumplir con transiciones estrictas de encendido y aceleración, respetando un tiempo mínimo de residencia en cada uno de los estados mencionados. Es importante destacar que, la operación de arranque sólo se puede llevar a cabo luego de que los equipos han permanecido en el modo "*Apagado*" (OFF) durante el tiempo de residencia mínimo establecido para dicho estado (3 horas en el ejemplo).

- **Procedimiento de apagado.** Es el procedimiento contrario al establecido para el arranque, es decir, una vez que los equipos han permanecido en modo "*Encendido*"

(ON) al menos el tiempo mínimo especificado, se puede iniciar el proceso de apagado, pasando por tres puntos de operación: "*Apagar compresor A - Fase I*", "*Apagar compresor A - Fase II*" y "*Apagar compresor B*". En este caso, la duración de cada modo de transición es de 1 hora.

- **Modo de operación *ON*.** Todo el equipamiento de la planta se encuentra en funcionamiento normal. Cabe destacar que, al cumplirse el tiempo mínimo de residencia en dicho estado, el equipamiento de la planta está en condiciones de atravesar una transición a otro modo de operación, de manera inmediata (pasando al modo "En espera") o siguiendo el procedimiento de apagado de equipos, detallado anteriormente (pasando al modo "OFF").

- **Modo de operación "En espera" o *ES*.** Como se mencionó, la planta puede pasar del modo ON al modo ES inmediatamente, si antes cumplió con el tiempo mínimo de permanencia. Del mismo modo, una vez que los equipos se encuentran en el modo ES, no pueden cambiar a otro estado hasta no haber cumplido con el respectivo tiempo preestablecido de estancia mínima. Generalmente, este modo se utiliza para paradas de planta cortas debido a restricciones de inventario o precios elevados del mercado de energía, evitando la detención y posterior encendido de los equipos en un tiempo relativamente acotado.

- **Modo de operación *OFF*.** Los equipos se apagan completamente, manteniéndose en este estado durante el tiempo mínimo necesario especificado. Tiene la misma lógica de funcionamiento que el modo de operación ON, pero de manera contraria.

A modo de resumen, el sistema bajo estudio presenta modos de operación con diferentes duraciones mínimas como requisito para un normal funcionamiento. Sin embargo, puede ocurrir que también se requiera cumplir con tiempos máximos de residencia en los tres grandes estados de permanencia (ON, OFF y ES), detallados anteriormente. Al mismo tiempo, se debe tener en cuenta que una transición entre modos de operación alternativos, es decir, un cambio de la planta de un punto de operación a

otro, puede suponer un consumo extra de energía eléctrica, dependiendo del caso. Por lo tanto, cualquier desviación de la operación esperada afectará a varios períodos de tiempo, tanto en niveles de producción como en costos energéticos.

Asimismo, para una representación completa del sistema productivo presentado, se deben considerar las tasas mínimas y máximas de producción en función de cada estado de operación, así como también la demanda diaria esperada y los precios de energía eléctrica por hora. Del mismo modo, es importante tener en cuenta, tanto la capacidad de almacenamiento de la planta, como el nivel mínimo de stock que debe tener el depósito al final del horizonte de planificación. Dicho nivel se establece con respecto al último período de tiempo programado, correspondiente al día de la semana.

Por último, como se mencionó con anterioridad, las operaciones de separación de aire se caracterizan por el continuo e intensivo consumo de energía eléctrica, por lo tanto, se definen niveles específicos de consumo para los diferentes modos de funcionamiento, que siguen una correlación lineal.

### 5.2.2. Mercados eléctricos

Dado que el costo de la energía es el componente principal del costo total de operación del proceso de producción detallado anteriormente, vale la pena discutir sobre los mercados existentes de energía eléctrica y el rol que tiene cada uno para este tipo de industrias. Tanto el precio, como la cantidad de energía disponible en el mercado, se definen de acuerdo al contrato de energía elegido. Por lo tanto, los procesos que presentan consumos intensivos deben implementar un plan de producción eficiente teniendo en cuenta los cambios en los precios dependientes del tiempo.

Las plantas CIEE, como la planta de separación de aire estudiada en este capítulo, puede participar en múltiples mercados de energía. Por ejemplo, los agentes del mercado energético pueden intercambiar contratos con períodos de entrega de distinta duración (anual, trimestral, mensual, etc.) mediante la firma de contratos (*mercado de contrato*), en

los que el precio y la cantidad constante de energía se fijan varios meses antes de la entrega. Al llegar el día D-1 (un día antes de la entrega/despacho físico de la energía de las centrales), los agentes intercambian energía para el día D en el mercado diario (ver Figura 5.2). Este contrato puede considerarse favorable, ya que el proveedor de energía puede ofrecer un precio más bajo a cambio de grandes cantidades de energía suministradas durante el tiempo acordado. Desde esta perspectiva, se necesita menos poder de equilibrio, se obtiene una red eléctrica más estable y es económicamente rentable para ambos agentes. Sin embargo, si la carga contratada no se consume por completo dentro de un período de tiempo, puede ser convenientemente ofrecida a la red. En la práctica, esto significa que la planta podría vender el excedente de electricidad en el *mercado spot* (o también llamado *mercado de desequilibrio*) cuando sea significativamente más rentable.

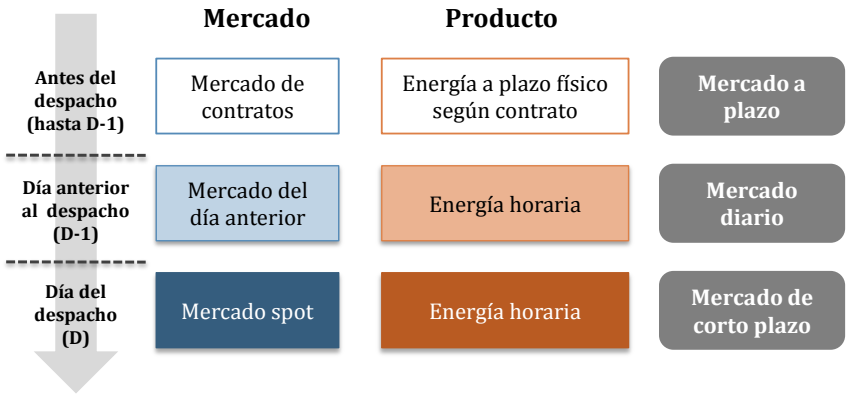



Figura 5.2. Secuencia de mercados de energía eléctrica.

Por otro lado, existe el *mercado del día anterior ("Day-ahead")*, donde se definen bloques de energía para el día siguiente en un horario específico del día, los cuales se compran en una subasta diaria. Los precios de la energía se conocen después de que la subasta se cierra, al mediodía del día anterior a la entrega. Por ejemplo, para un horizonte de tiempo semanal, de lunes a domingo, el domingo se conoce el precio y la cantidad de energía para cada hora del día lunes, mientras que se desconoce dicha información para los días restantes (de martes a domingo). Sin embargo, se puede contar con pronósticos de precios para los siguientes días (por ejemplo, para los siguientes 9 días) en el mercado diario.

Por último, la energía eléctrica no contratada resultante de los desequilibrios en la red de energía física y por otro lado, el intento de los operadores para que coincidan la oferta y la demanda, determinan el ya mencionado *mercado de desequilibrio* o *mercado spot* (en tiempo real). En este mercado se compra energía al precio horario, el cual se conoce 15 minutos después de que se consume la energía. El desafío es predecir el tiempo durante el cual el precio permanecerá rentable, para que la planta tenga tiempo de implementar políticas de reacción, o incluso para predecir los picos en la curva de precios (Pedro M Castro et al., 2009). Como ejemplo, la Figura 5.3 ilustra los perfiles de precios de energía correspondientes a un mercado spot. La línea negra muestra los precios conocidos, mientras que la curva roja es el pronóstico de precios. Las curvas en color celeste reflejan el ruido de los pronósticos conforme avanza el tiempo, mostrando una mayor imprecisión en los últimos días.

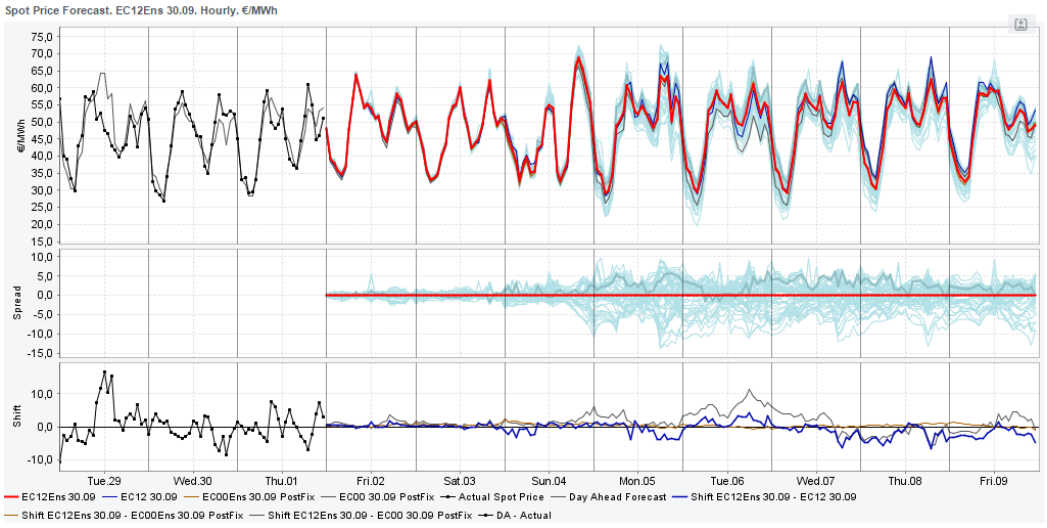



Figura 5.3. Ejemplo de esquema de precios de la energía en el mercado spot.

Se supone que la planta de separación de aire bajo estudio puede comprar energía en el mercado diario, donde las nominaciones se deciden a partir de la producción esperada para el día siguiente, y que dispone de un pronóstico de los precios de la energía de los próximos 7 días para el mercado diario, es decir, siempre cuenta con el pronóstico de precios de una semana, el cual se va actualizando según avancen los días.

## 5.3. Red de transición de estados del proceso (PSTN)

Un aspecto importante del problema descripto en la sección 5.2.1, es la restricción operativa sobre la cantidad mínima de tiempo que la planta debe permanecer en un mismo modo de operación (Figura 5.1), la cual se debe cumplir de manera estricta. Esto significa que, un desvío en la programación de la planta puede llegar a afectar varios períodos de tiempo.

Con el objetivo de representar de forma sistemática los estados de transición del sistema bajo estudio, considerando además, secuencias de operaciones predefinidas y tiempos mínimos y máximos de permanencia en cada uno de dichos estados, se propone una nueva formulación de modelado explícito de transiciones factibles de una planta CIEE, llamada "*Process State Transition Network*" (PSTN). La Figura 5.4 muestra el concepto desarrollado, donde se denomina "estado" operacional o de transición a un posible punto de operación de la planta. En el esquema cada nodo (denotado por rectángulos) representa un estado de transición específico del ciclo de separación del aire. Por otro lado, las transiciones factibles entre dichos estados están representadas por arcos dirigidos, que reflejan la dinámica del proceso. Tanto los nodos como los arcos implican restricciones operativas que deben cumplirse en todo momento.

El concepto PSTN permite, como se puede observar en la Figura 5.4, la desagregación de los modos de operación y un modelado más detallado del comportamiento del proceso durante una transición. Por ejemplo, los estados con mayor duración mínima (ON, OFF, ES) se descomponen en 3 sub-estados de 1 hora, cada uno definido como "*Estado de transición inicial*", "*Estado de transición intermedio*" y "*Estado de transición crítico*", respectivamente. En el ejemplo de la Figura 5.1, estos modos de operación deben cumplir con una estancia mínima de 3 horas, lo que equivaldría a pasar por los tres estados mencionados. En el caso que la planta deba operar por un período mayor a 3 horas, debe pasar por los estados de transición inicial e intermedio (1 hora en

cada uno), y permanecer en el estado de transición crítico ($ON_n$, $OFF_n$ o $ES_n$, según corresponda) hasta cumplir dicho período de permanencia.

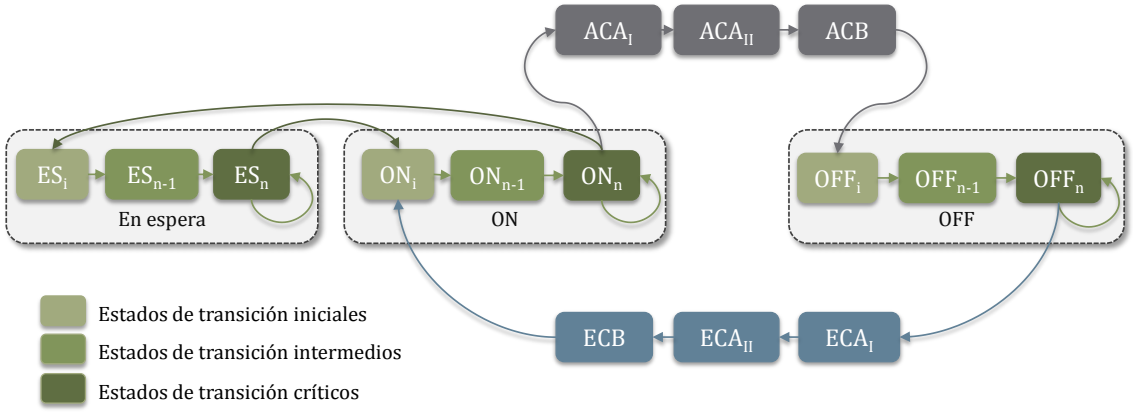



Figura 5.4. Esquema de transición de estados PSTN.

En resumen, la principal contribución de la metodológica desarrollada es la desagregación del proceso de transición de un estado de operación a otro en un determinado tiempo. Esta desagregación resulta favorable al representar el problema de programación de las operaciones de una manera conceptualmente diferente, reduciendo las variables binarias y definiendo restricciones más simples, con respecto a los modelos anteriormente desarrollados, para este tipo de sistemas.

## 5.4. Formulación matemática

Para la resolución del problema de programación de la producción planteado en este capítulo, se desarrolla un modelo matemático lineal mixto-entero basado en el concepto PSTN presentado anteriormente (llamado modelo PSTN-MILP). Este modelo intenta dar solución a un proceso productivo que involucra diferentes estados operacionales, cada uno de los cuales incurre en un costo de energía eléctrica, es decir, influye directamente en el principal costo operacional. El objetivo es presentar una herramienta eficiente que permita obtener el programa de operaciones de una planta con

elevados niveles de consumo de energía, que minimice el costo total de energía en un horizonte temporal específico.

La formulación MILP propuesta está basada en una representación discreta del tiempo. En la literatura se pueden encontrar diferentes modelos matemáticos basados en tiempo continuo, utilizados eficientemente para la resolución de múltiples problemas de programación de operaciones con restricciones de energía eléctrica (Castro et al., 2011b; Hadera et al., 2016, 2015; Haït y Artigues, 2011; Nolde y Morari, 2010). Sin embargo, dadas las restricciones operativas del problema abordado en esta tesis y la característica fluctuante de los precios de la energía, según el mercado eléctrico donde participa, se utiliza el enfoque basado en tiempos discretos. Por lo tanto, el horizonte de planificación se divide en $T$ intervalos de tiempo fijos de igual duración, es decir, con $\Delta t$ constante. Cada uno de ellos representa un período de tiempo de una duración de una hora. De acuerdo con esta discretización del tiempo, si se considera un horizonte de planificación de una semana, el cual consta de 168 horas, el conjunto se define como $T = \{1,2,\dots,T^{last}\}$, donde $T^{last} = 168$.

Bajo las condiciones mencionadas, todas las variables del modelo estarán definidas con sus índices de período $t$. Asimismo, los diferentes sucesos del sistema (inicio y fin de modos de operación, transiciones entre diferentes estados operacionales de la planta, cambios en la tasas de producción, etc.) pueden ocurrir únicamente en los límites de los intervalos de tiempo, ya que durante el transcurso de un determinado período, la planta se encuentra en un mismo estado, operando bajo las mismas condiciones. En particular, los cambios en el sistema productivo pueden ocurrir en el inicio de cada período $t$. A modo de ejemplo, en la Figura 5.5 se muestra cómo el equipamiento que conforma la planta se encuentra apagado durante las primeras 4 horas, para luego llevar a cabo el procedimiento de encendido durante las siguientes 3 horas, llegando a un normal funcionamiento a las 8 horas de comenzar la semana. Con el fin de cumplir con la estancia mínima establecida para el modo ON, la planta debe operar en dicho modo hasta la hora 10 (siguiendo el esquema de la Figura 5.1, de 3 horas).

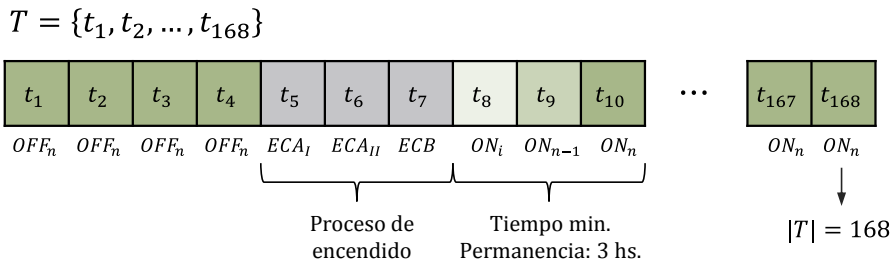



Figura 5.5. Discretización del horizonte de tiempo.

A continuación, se describe el modelo matemático propuesto y el conjunto de restricciones operativas que se tienen en cuenta para minimizar el costo total asociado al consumo de energía. Luego, se presenta un modelo alternativo propuesto por Mitra et al. (2012), el cual se adapta al problema bajo estudio y se utiliza posteriormente para analizar diferentes ejemplos y evaluar el desempeño del modelo presentado en este capítulo.

### 5.4.1. Modelo MILP basado en tiempo discreto

El sistema productivo estudiado, como se ha indicado, participa en mercados de energía donde el precio varía de un momento a otro. Por lo tanto, experimenta un comportamiento dinámico en el tiempo (Artigues et al., 2013; Karwan y Keblis, 2007; Mitra et al., 2014a, 2014b, 2013; Zamarripa et al., 2016; Q. Zhang et al., 2016b). Este comportamiento se representa a través de restricciones que se deben cumplir de manera estricta. Por ejemplo, cuando los equipos deben efectuar procedimientos que involucran transiciones específicas, estos deben permanecer en determinadas condiciones durante un intervalo de tiempo determinado, y al mismo tiempo, deben cumplir con los niveles de producción para satisfacer la demanda y los niveles de inventario requeridos.

Para un correcto modelado, se debe tener en cuenta que existen diferentes tipos transiciones:

- ✓ transiciones entre diferentes estados operativos
- ✓ transiciones en un mismo estado

A fin de identificar estas transiciones, se podría suponer que una buena decisión de operación es que la planta permanezca apagada o en espera (estado $OFF_n$ o $ES_n$, respectivamente), según sea más conveniente, durante los períodos donde los precios de energía son altos, siempre y cuando se puedan satisfacer las restricciones de la demanda y los niveles mínimos de inventario. Suponiendo que permanece en el estado $OFF_n$ durante un tiempo determinando (y cumpliendo, además, con el tiempo mínimo de residencia requerido), es decir, se efectúan *transiciones en un mismo estado*, y que luego, los precios presentan un descenso, es lógico pensar que la planta debe comenzar a producir y aprovechar los beneficios que esto conlleva. De este modo, al llevar a cabo el proceso de encendido de los equipos, se debe efectuar la transición de los estados $ECA_I, ECA_{II}, ECB$, con lo cual se producen *transiciones entre diferentes estados operativos*. Una vez que los equipos se encuentren en funcionamiento normal ($ON_n$) y cumplieron con el tiempo requerido de permanencia en dicho estado, pueden permanecer en la condición o el comenzar con las transiciones de apagado ($ACA_I, ACA_{II}, ACB$), o cambiar al modo ES (en espera).

A continuación, se presentan los conjuntos de restricciones que integran la formulación MILP propuesta. Los mismos quedan definidos en función de cinco categorías:

(i) Restricciones sobre los estados operacionales y las transiciones permitidas para el normal funcionamiento de la planta.

(ii) Restricciones de tiempos mínimos y máximos de permanencia permitidos para cada tipo de estado.

(iii) Restricciones de producción, donde se tienen en cuenta las tasas de producción y los niveles de inventario mínimos y máximos.

(iv) Restricciones sobre el consumo de energía en función de los pronósticos de tiempo utilizados, tanto para la demanda esperada, como para los precios de energía en el mercado diario.

(v) Restricción de la función objetivo.

#### 5.4.1.1 Restricciones de estados y transiciones operacionales

Las restricciones que se presentan a continuación permiten representar los estados de operación $s \in S$ y los procesos secuenciales de transición entre dichos estados a través de la variable binaria $W_{s,t}$, determinando el estado de operación $s \in S$ de la planta en cada período $t \in T$ del horizonte de planificación.

- ✓ $W_{s,t}$: variable binaria que representa el estado $s$ del equipamiento de la planta en cada período de tiempo $t$

**Asignación de estados de operación.** La restricción (5.1) establece que a cada hora $t$ la planta de operar en un único estado $s$.

$$\sum_{s} W_{s,t} = 1 \qquad \forall t \in T \tag{5.1}$$

**Estados de transición secuencial.** El cambio de la planta, de un punto de operación a otro, corresponde a una decisión discreta asociada a un período o a un conjunto de períodos de tiempo. De acuerdo con el concepto PSTN, presentado en la sección 5.3, las transiciones entre los estados pueden ocurrir siempre que se ejecuten en el orden correcto. En otras palabras, hay secuencias predefinidas de estados de operación que describen el comportamiento de conmutación de los equipos de la planta. Para su correcto modelado se definen 4 subconjuntos de estados, asociados a los modos de operación $m \in M$, donde $M = \{ON, OFF, ES\}$ (ver Figura 5.4):

- $S^{initial}$: estados que modelan el inicio de los modos de operación $m$ $(ON_i, OFF_i, ES_i)$.
- $S^{inter}$: estados correspondientes a la transición intermedia de los modos $m$ $(ON_{n-1}, OFF_{n-1}, ES_{n-1})$.

- $S^{critical}$: estados que representan la transición crítica, la cual se puede repetir de manera consecutiva durante un período de tiempo determinado $(ON_n, OFF_n, ES_n)$.

- $LIC_s$: estados que preceden inmediatamente a un estado crítico $s \in S^{critical}$. Por ejemplo, para $s = ON_n$: $LIC_{ON_n} = \{ON_{n-1}, ON_n\}$.

La secuencia de operación que se debe cumplir cuando se inician los estados de funcionamiento normal, de apagado y de espera, es representada por las restricciones (5.2) y (5.3). Si la planta se encuentra en el estado de transición inicial $s \in S^{initial}$ del modo ON, OFF o ES en el período $t$, entonces en el tiempo $t+1$ y $t+2$ deberá operar en los estados de transición intermedios $s \in S^{inter}$ y críticos $s \in S^{critical}$ correspondientes, respectivamente. Estas relaciones quedan representadas en la Figura 5.6: (i) con línea de punto naranja la relación de precedencia entre los estados iniciales e intermedios, correspondiente a la restricción (5.2), y (ii) con línea de puntos azul la relación de los últimos con los estados críticos, representada por (5.3).

$$W_{s,t} = \sum_{s' \in S^{inter}} W_{s',t+1} \qquad \forall t \in T, s \in S^{initial} \tag{5.2}$$

$$W_{s,t} \leq W_{s',t+1} \qquad \forall t \in T, s \in LIC_s, s' \in S^{critical} \tag{5.3}$$

Es importante destacar que, los procedimientos de arranque y apagado de todos los equipos no se pueden interrumpir. Durante estos procesos la planta debe cumplir con determinadas secuencias de transición de estados, atravesando diferentes puntos operacionales que presentan el mismo tiempo de duración (1 hora). Por ejemplo, si los equipos se encuentran apagados, no pueden ser encendidos directamente para operar de manera normal de un momento a otro. Este cambio de estado requiere un proceso que abarca varios períodos de tiempo; en el caso del ejemplo, los equipos deben atravesar los estados: $ECA_I$ , $ECA_{II}$ y $ECB$ . Las secuencias factibles $off \rightarrow arranque \rightarrow on$ y

$on \rightarrow apagado \rightarrow off$, se satisfacen a través de las restricciones (5.4) y (5.5), respectivamente. Notar que, los parámetros $nd$ y $ns$ utilizados representan el número de estados intermedios en el proceso de apagado y de encendido de los equipos de la planta. Además, para la definición de las restricciones, se definen nuevos subconjuntos de estados para modelar dichas secuencias:

- $S^{down-initial}$, $S^{up-initial}$: estados que dan inicio al proceso de apagado y encendido de la planta, respectivamente.
- $S^{down-inter}$, $S^{up-inter}$: estados asociados a las transiciones intermedias presentes en los procedimientos de apagado y encendido de la planta, respectivamente.

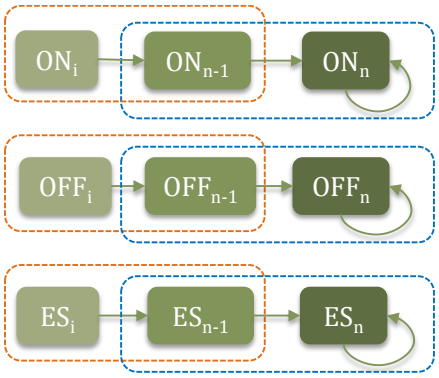



Figura 5.6. Relación de los estados de los modos de operación globales $m \in M$.

$$nd * W_{s,t} = \sum_{s' \in S^{down-inter}} \sum_{t'=t+1}^{t'=t+nd} W_{s',t'} \qquad \forall t \in T, s \in S^{down-initial} \tag{5.4}$$

$$ns * W_{s,t} = \sum_{s' \in S^{up-inter}} \sum_{t'=t+1}^{t'=t+ns} W_{s',t'} \qquad \forall t \in T, s \in S^{up-initial} \tag{5.5}$$

Sin embargo, son necesarias restricciones adicionales para completar los procesos de encendido y apagado de equipos. Las restricciones (5.6) y (5.7) modelan la última transición de estado requerida para terminar cada proceso, respectivamente.

$$W_{s,t} = W_{s',t+1} \qquad \forall t \in T, s \in S^{down-initial}, s' \in S^{initial}: s' = OFF_i \tag{5.6}$$

$$W_{s,t} \leq W_{s',t+1} \qquad \forall t \in T, s \in S^{up-initial}, s' \in S^{initial}: s' = ON_i \tag{5.7}$$

Como se puede observar en la Figura 5.4, y con mayor detalle en la Figura 5.7, el primer estado del modo ON ($ON_i$) tiene dos posibles estados predecesores ($ES_n$ y $ECB$), a diferencia del resto de los estados que sólo tienen uno. Esto se ve reflejado en el uso de una igualdad en la restricción (5.6) y de desigualdad en la restricción (5.7). Por esta razón, también es necesario definir la restricción (5.8), representando los dos posibles caminos previos que pueden tener los equipos antes de alcanzar el estado $ON_i$. Notar que, esta restricción se puede adaptar fácilmente para ser aplicada a cualquier estado que presente dos predecesores inmediatos.

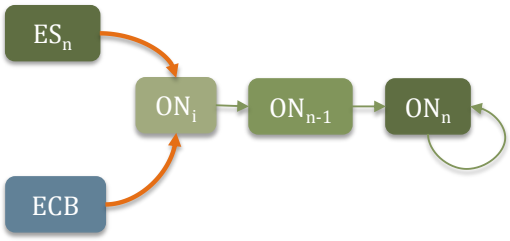



Figura 5.7. Predecesores del estado inicial ($ON_i$) del modo de operación ON.

$$W_{s,t} + W_{s',t} \geq W_{s'',t+1} \tag{5.8}$$

$$\forall t \in T, s \in S^{critical}, s' \in S^{up-initial}, s'' \in S^{initial}: s = ES_n, s'' = ON_i$$

**Estados de transición críticos.** Cuando la planta opera en un estado crítico ($ON_n$, $OFF_n$ o $ES_n$) en un determinado período $t$, puede permanecer en el mismo estado o cambiar a otro en el siguiente período $(t+1)$, es decir, puede dar lugar a una transición en el mismo estado o a otro diferente. La Figura 5.8 muestra un ejemplo de las posibles transiciones entre diferentes estados y en el mismo estado, teniendo como punto de

partida el estado crítico del modo $m = ON$: $ON_n \rightarrow ACA_I$, $ON_n \rightarrow ES_i$ y $ON_n \rightarrow ON_n$. Notar que, las transiciones descriptas pueden ocurrir sólo en los estados críticos. La restricción (5.9) describe estas posibles transiciones, para lo cual se define un nuevo subconjunto:

- $NTS_s$: subconjunto de estados en los que puede operar la planta, inmediatamente después de haber permanecido en el estado crítico $s$ el tiempo requiero $mn$.

$$W_{s,t} + W_{s',t} = \sum_{s'' \in NTS_s} W_{s'',t+1} \qquad \forall t \in T, s \in S^{critical}, s' \in LIC_s \tag{5.9}$$

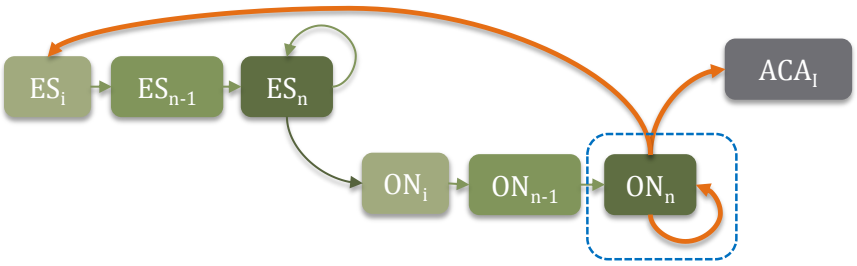



Figura 5.8. Posibles transiciones a partir del estado crítico $ON_n$.

Se puede observar que, debido a que la planta debe operar en un sólo estado en cada hora (restricción 5.2), sólo se puede activar una variable binaria en cada lado de la igualdad (5.9), satisfaciendo todas las transiciones factibles. Cabe destacar que, dicha restricción también se satisface para la transición de un estado intermedio ($ON_{n-1}$, $OFF_{n-1}$ o $ES_{n-1}$) a su correspondiente estado crítico.

### 5.4.1.2 Restricciones de tiempo de permanencia

La planta tiene restricciones de residencia mínima y/o máxima en los diferentes modos operacionales $m$, por consiguiente se deben imponer límites inferiores y superiores en el tiempo de permanencia en determinados estados, según corresponda. En particular, estas restricciones se activan cuando se inicia alguno de los tres modos operacionales globales, es decir, luego de realizar el procedimiento de encendido de los equipos, cuando se requiere que queden en espera o luego del proceso de apagado. Como se describió en la

sección 5.3, cada uno de estos modos se descomponen en tres estados o puntos de operación $s$, de los cuales, el clasificado como *estado crítico* ($s \in S^{critical}$) es el que se puede mantener con una duración que involucre más de un período de tiempo $t$. Asimismo, vale la pena destacar que, el resto de los estados requieren una permanencia estricta de 1 hora de operación, equivalente a la menor unidad de tiempo en que se divide el horizonte de planificación. Por lo tanto, se aplican las restricciones (5.10)-(5.13) a los modos mencionados, para modelar las transiciones en un mismo estado, garantizando la permanencia de los equipos de la planta durante el intervalo de tiempo requerido.

**Tiempo mínimo de permanencia.** Como se puede observar en la Figura 5.1, los modos de operación globales $m$ deben cumplir con un tiempo de permanencia mínimo de 3 horas, por lo tanto, una transición hacia uno de ellos afectará a varios períodos de tiempo. Notar que, debido a que para otra configuración de planta este valor podría variar, se define un parámetro que refleja este dato: $min_{res}$, y se introducen las siguientes restricciones:

$$mn * W_{s,t} = \sum_{s' \in S^{critical}} \sum_{t'=t+1}^{t+mn} W_{s',t'} \qquad \forall t \in T, s \in S^{inter} : t \leq (FT - mn) \tag{5.10}$$

$$mn = min_{res} - 2 \tag{5.11}$$

El parámetro $mn$ utilizado en ambas restricciones representa la cantidad de horas que la planta debe permanecer en los estados críticos (últimos sub-modos de los modos globales), produciéndose transiciones en el mismo estado. Se calcula restándole 2 a el valor de $min_{res}$, ya que se debe descontar el tiempo requiero por los equipos para atravesar los estados iniciales e intermedios, con duración fija de 1 hora cada uno. De este modo, los estados críticos se repetirán $mn$ horas como mínimo. Sólo entonces, la configuración del sistema puede pasar del estado de transición crítico $s \in S^{critical}$ a cualquiera de los estados que pertenecen al conjunto $NTS_s$.

**Tiempo máximo de permanencia.** Al igual que se definió para la restricción anterior, pero en este caso, considerando el máximo tiempo permitido de residencia de los equipos en un mismo estado, se presentan las restricciones (5.12) y (5.13), donde se calcula el número máximo de horas $max_{res}$ que la planta puede operar en los modos globales, tales como encendido, apagado o estados en espera. Por lo tanto, no puede permanecer más de $mx$ períodos de tiempo en cualquier estado crítico de transición.

$$\sum_{t'=t}^{t+mx} W_{s,t'} \leq mx \qquad \forall t \in T, s \in S^{critical}: t \leq \left(T^{last} - max_{res}\right) \tag{5.12}$$

$$mx = max_{res} - 2 \tag{5.13}$$

### 5.4.1.3 Restricciones de producción

El conjunto de restricciones que se presenta a continuación define las tasas de producción, los niveles de inventario y la relación entre los mismos a través de balances de masa, para lo cual se utilizan las siguientes variables continuas:

- ✓ $P_{s,t}$: variable que determina la tasa de producción de los equipos de la planta en cada período de tiempo $t$
- ✓ $I_t$: variable que representa la cantidad de producto que tiene en inventario la planta al inicio de cada período $t$.

**Tasas de producción.** Cada estado de operación tiene asociado un nivel o límite de producción mínimo $MinP_s$ y máximo $MaxP_s$. A través de la restricción (5.14) se utiliza la variable binaria $W_{s,t}$ para garantizar que la tasa de producción $P_{s,t}$ en cada periodo $t$, siempre se encuentre dentro de los límites preestablecidos.

$$W_{s,t} * MinP_s \leq P_{s,t} \leq W_{s,t} * MaxP_s \qquad \forall t \in T, s \in S \tag{5.14}$$

Asimismo, se debe considerar que la velocidad de producción no pueden cambiar abruptamente de un período a otro, es decir, la producción debe variar gradualmente a lo largo del tiempo, tanto cuando se incrementa, como cuando disminuye. Por ejemplo, si la planta comienza el proceso de puesta en marcha de los equipos en el período $t$, recién puede comenzar a producir en el periodo $t + 3$ (es decir, cuando $W_{ON_i,t+3} = 1$), sin embargo, no lo puede llevar a cabo con su tasa máxima de producción. Para alcanzar la máxima velocidad de producción, debe operar un mínimo períodos consecutivos en el estado $ON$. De este modo, se toma como hipótesis que el equipamiento comienza a operar en estado activado ($ON_i$) con el nivel mínimo de producción ($MinP_s$), a través de la restricción (5.15), y luego, éste puede variar según sea necesario. Para modelar dicha restricción se define un subconjunto de estados con producción mínima: $S^{minProd}$.

Asimismo, para representar completamente el comportamiento descripto, se definen los parámetros *inc* y *dec* con el fin de representar el porcentaje en que se puede incrementar o disminuir la tasa de producción cuando ocurre una transición de un estado a otro. Las restricciones (5.16) y (5.17) describen dicho comportamiento.

$$P_{s,t} = W_{s,t} * MinP_s \qquad \forall t \in T, s \in S^{minProd} \tag{5.15}$$

$$P_{s',t-1} * dec \leq P_{s,t} \leq P_{s',t-1} * inc \qquad \forall t \in T, s' \in S^{initial}, s \in S^{inter} \tag{5.16}$$

$$\left(P_{s'',t-1} + P_{s',t-1}\right) * dec \leq P_{s,t} \leq \left(P_{s'',t-1} + P_{s',t-1}\right) * inc \tag{5.17}$$

$$\forall t \in T, s' \in S^{inter},\ s \in S^{critical}, s'' \in S^{initial}$$

**Niveles de inventario.** Se realiza, para cada periodo de tiempo $t$, un balance de masa entre la cantidad de producto que se encuentra en inventario ($I_t$), la cantidad producida ($P_{s,t}$) y la cantidad de producto demandado ($ED_t$), formulando las restricciones (5.18) y (5.19). La primera de ellas calcula el inventario en la primera hora del horizonte ($t = 1$), considerando el inventario inicial del horizonte de planificación que se tiene como dato ($I_0$), mientras que la segunda considera los períodos restantes ($t > 1$), teniendo en cuenta el nivel de inventario de la hora anterior ($I_{t-1}$).

$$I_t = I_0 + \sum_s P_{s,t} - ED_t \qquad \forall t \in T: t = 1 \tag{5.18}$$

$$I_t = I_{t-1} + \sum_s P_{s,t} - ED_t \qquad \forall t \in T: t > 1 \tag{5.19}$$

Por otro lado, existe una cantidad mínima y máxima del inventario que debe cumplirse en todo momento, representada por los parámetros $Qmin$ y $Qmax$, respectivamente. Esta condición se modela mediante la siguiente restricción:

$$Qmin \leq I_t \leq Qmax \qquad \forall t \in T \tag{5.20}$$

**Nivel de inventario final.** Por último, la planta debe cumplir con un nivel mínimo de inventario al final del horizonte de planificación, correspondiente al último día de la semana ($T^{last} = 168$). Este valor es denotado por el parámetro $MDTL_{T^{last}}$, que representa específicamente la cantidad mínima de producto que se debe almacenar al final de la semana de operación, necesaria para seguir operando la semana siguiente. De esta manera, el nivel de inventario para $T^{last}$ se calcula como sigue:

$$I_t \geq MDTL_{T^{last}} \qquad \forall t \in T^{last} \tag{5.21}$$

#### 5.4.1.4 Restricciones de consumo de energía

Para estimar la cantidad de energía eléctrica consumida en cada período $t$, se define la variable continua:

- ✓ $PW_t$: variable que determina la potencia total consumida en el período de tiempo $t$.

Además, se define la restricción (5.22) para calcular la potencia total, la cual está compuesta por un componente fijo $FPC_s$ y uno variable $VPC_s$ de consumo de energía eléctrica. El primer término de la restricción está asociado cada punto de operación $s$ en que se encuentren los equipos en el momento $t$, mientras que el segundo depende de la tasa de producción con que opere la planta en dicho período y para el mismo estado.

$$PW_t = \sum_s (W_{s,t} * FPC_s + VPC_s * P_{s,t}) \qquad \forall t \in T \tag{5.22}$$

#### 5.4.1.5 Función objetivo

La función objetivo del modelo MILP propuesto, como ya se ha mencionado, es minimizar el costo total de la energía y queda representada mediante la restricción (5.23). Para este cálculo se utiliza la potencia total consumida, previamente presentada en la restricción (5.22), como factor fundamental. Cabe destacar que, el pronóstico del precio de la energía puede ser único (precio fijo durante todo el horizonte de planificación, $EP^{FIXED}$) o variar en el tiempo (por hora o por día, $EP_t$).

$$\text{Min } Cost = \sum_t (PW_t * EP_t) \tag{5.23}$$

## 5.4.2. Modelo alternativo

Un enfoque alternativo para la resolución del mismo problema fue propuesto por Mitra et al. (2012), quienes presentan un modelo MILP para optimizar la programación de las operaciones de producción para procesos continuos con un uso intensivo de energía, similares al presentado en la sección 5.2. Al igual que el modelo PSTN-MILP desarrollado en esta tesis, Mitra et al. (2012) propone una formulación matemática de tiempo discreto, que permite determinar los niveles de producción e inventario y los modos de operación para cada período de tiempo, de acuerdo con esquemas de precios de electricidad dependientes del tiempo. Del mismo modo, dicho modelo se utiliza para evaluar un caso de estudio industrial de una planta PSA que produce múltiples productos líquidos y gaseosos, como oxígeno, nitrógeno y argón.

La principal diferencia entre el modelo de Mitra et al. (2012) y el modelo PSTN-MLIP es la representación de los puntos de operación de la planta y por lo tanto, la forma de capturar el comportamiento de los modos de transición. Mientras que el primero representa las transiciones operacionales que definen los modos de transición globales $m$ como un conjunto ($m \in M$) y captura el comportamiento de transición de la planta durante un período de tiempo específico, el segundo sigue el concepto PSTN desagregando estos modos de operación en sub-modos operativos (llamados estados de operación) para cada período de tiempo.

Con el objetivo de llevar a cabo una posterior comparación detallada entre ambos modelos y poder evaluar la eficiencia computacional del modelo PSTN-MILP desarrollado, se presenta el modelo mencionado de la literatura, el cual se adapta a las características del problema descripto en este capítulo.

Para modelar los niveles de producción de una planta de separación de aire, Mitra et al. (2012) utilizan una variable binaria $Y_{p,m}^{t}$, equivalente a la variable $W_{st}$ utilizada en el modelo PSTN-MILP:

- ✓ $Y_{p,m}^{t}$ : variable que toma valor 1 cuando la planta $p$ opera en el modo $m$ durante el período $t$.

Sin embargo, la diferencia más importante entre ambos modelos es que, tanto Mitra et al. (2012) como posteriormente Mitra et al. (2013), utilizan una variable binaria adicional de precedencia de modos de operación $Z_{m',m}^{t}$ , la cual aumenta considerablemente el tamaño del modelo, y por lo tanto la complejidad.

- ✓ $Z_{m',m}^{t}$: variable binaria que representa la transición que ocurre desde el período de tiempo $t-1$ a $t$, pasando del modo $m'$ al modo $m$.

Para modelar posibles transiciones de un período $t-1$ a $t$ se definen las restricciones (5.24) y (5.25), donde si se cumple que $Y_{m}^{t-1}=1$ y $Y_{m'}^{t}=1$, entonces se debe cumplir la transición de $m$ a $m'$, es decir, $Z_{m,m'}^{t}=1$. Posteriormente, con el objetivo de reducir el número de restricciones utilizadas, Mitra et al. (2013) las reformulan, obteniendo la restricción (5.26).

$$\sum_{m' \in M} Z_{m',m}^{t} = Y_{m}^{t} \qquad \forall t \in T, m \in M \tag{5.24}$$

$$\sum_{m' \in M} Z_{m,m'}^{t} = Y_{m}^{t-1} \qquad \forall t \in T, m \in M \tag{5.25}$$

$$\sum_{m' \in M} Z_{m',m}^{t} - \sum_{m' \in M} Z_{m,m'}^{t} = Y_{m}^{t} - Y_{m}^{t-1} \qquad \forall t \in T, m \in M \tag{5.26}$$

Por otra parte, para modelar el tiempo de residencia mínima de los modos operacionales, Mitra et al. (2013) presentan una modificación (5.27) de la restricción introducida por Mitra et al. (2012). Además, introducen la restricción (5.28).

$$Y_{m'}^{t} \geq \sum_{\theta=0}^{K_{m,m'}^{min}-1} Z_{m,m'}^{t-\theta} \qquad \forall (m,m') \in Seq, t \in T \tag{5.27}$$

$$Z_{m,m''}^{t-K_{m,m''}^{min}} - Z_{m'',m'}^{t} = 0 \qquad \forall t \in T, (m,m',m'') \in Seq \tag{5.28}$$

Notar que, $K_{m,m'}^{min}$ representa el tiempo mínimo de permanencia que se debe cumplir al pasar del modo $m$ a $m'$, y es equivalente a $min_{res}$ para ON, OFF y ES. Además, se define el conjunto $Seq$, como las posibles transiciones desde el modo de producción $m$ hasta $m'$, pasando por $m''$.

Además, incorporan la variable continua $Pr^t$ para calcular los niveles de producción total en cada hora $t$ a través de la siguiente restricción:

$$Pr^t = \sum_{m \in M} \overline{Pr}_m^t \qquad \forall t \in T \tag{5.29}$$

donde $\overline{Pr}_m^t$ representa, al igual que $P_{s,t}$, la tasa de producción de cada punto operacional de la planta en cada período de tiempo. Del mismo modo, los parámetros $MaxP_s$ y $\overline{M}_m$ (Mitra et al., 2012), y las restricciones (5.14) y (5.25), respectivamente, son equivalentes. Notar que, al considerar una única planta PSA, la variable $Y_{p,m}^t$ se puede representar como $Y_m^t$.

$$\overline{Pr}_m^t \leq \overline{M}_m * Y_m^t \qquad \forall m \in M, t \in T \tag{5.30}$$

Por otro lado, se define un conjunto para representar las transiciones prohibidas entre los modos de operación $m$ y $m'$ ($DAL$), el cual se utiliza en la restricción (5.31) para modelar dichas transiciones.

$$Z^{t}_{m,m'} = 0 \qquad \forall t \in T, (m, m') \in DAL \tag{5.31}$$

Finalmente, se modela el balance de masa de la planta mediante las restricciones (5.20) y (5.32), donde se puede observar que esta última es equivalente a la restricción (5.19) presentada para el modelo PSTN-MILP.

$$I_t + Pr^t = I_{t+1} + ED_t \qquad \forall t \in T \tag{5.32}$$

## 5.5. Aplicación de la estrategia de solución a un caso de estudio

En esta sección se presentan diferentes instancias de un caso de estudio real, con el fin de resolver el problema de programación de las operaciones de una planta industrial de separación de aire e ilustrar, tanto el uso, como el desempeño, que ofrece el enfoque PSTN-MILP propuesto en este capítulo. De esta manera, se modelan diferentes valores de los parámetros de entrada para un horizonte de planificación semanal (168 horas): pronósticos de precios de la energía, configuraciones de planta y demandas esperadas.

Las instancias del problema se implementan y resuelven en el software GAMS 24.9.2 con un procesador Intel Xeon X5650, 2.6 GHz y 32 GB de RAM. Para resolver los modelos MILP se utilizan los optimizadores CPLEX 12.7.1.0 y GUROBI 6.5.2, con diferentes valores de optimalidad (0 %, 1 % y 5 %) como criterios de terminación, de manera de poder evaluar la eficiencia computacional de las formulaciones matemáticas.

A continuación, se presentan los datos industriales reales utilizados para generar las instancias del problema y los resultados estadísticos obtenidos. Por un lado, se realiza una comparación entre el modelo de optimización PSTN-MILP y el modelo alternativo desarrollado por Mitra et al. (2012), de manera de determinar la eficiencia del enfoque propuesto, en términos de esfuerzo computacional requerido. Por último, se evalúa la aplicación de dicho enfoque y la eficiencia del plan detallado obtenido para diferentes

configuraciones de planta, tanto en términos del consumo energético como del costo asociado.

### 5.5.1. Caso de estudio

La aplicabilidad del modelo PSTN-MILP propuesto en problemas industriales reales se testea a través de la resolución de un caso de estudio correspondiente, como se ha mencionado, a una planta PSA. El ejemplo trata sobre un sistema que emplea un equipamiento que atraviesa diferentes estados de operación, los cuales tienen asociados niveles de producción y consumos de energía específicos. En la Tabla 5.1 se muestran los principales datos de cada estado operacional. Como se puede apreciar, se presentan, tanto los valores mínimos y máximos de producción, como los consumos fijos y variables de energía eléctrica. Estos últimos corresponden a los valores de los parámetros $FPC_s$ y $VPC_s$, respectivamente, y se utilizan para calcular la potencia consumida en cada período, a través de la restricción (5.22).

Tabla 5.1. Tasas de producción y consumo de energía para cada modo de operación.

| | | Modo On | Modo Off | Modo ES | Apagado Compresor A | | Apagado Compresor B | Encendido Compresor A | | Encendido Compresor B |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Fase I | Fase II | | Fase I | Fase II | |
| Producción mínima unidades/hora | | 0,8 | 0 | 0 | 0,8 | 0,8 | 0,8 | 0 | 0 | 0 |
| Producción máxima unidades/hora | | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Consumo de energía* MWh/unidad | $FPC_s$ | 0 | 0 | 1,1 | 0,63 | 0,7 | 2,41 | 0,36 | 0,6 | 3,1 |
| | $VPC_s$ | 11,25 | 0 | 0 | 11,25 | 11,25 | 11,25 | 0 | 0 | 0 |

*Función lineal del consumo de energía: $PW_t = \sum_s (W_{st} * FPC_s + VPC_s * P_{st}), \forall t \in T$.

La Tabla 5.2 resume la información relativa a la demanda diaria esperada para el horizonte de planificación de una semana. Además, se debe considerar que el almacenamiento en la planta puede variar entre 34 y 87 unidades, valores representados por los parámetros $Qmin$ y $Qmax$, respectivamente.

Tabla 5.2. Demanda esperada para una semana (unidad/día).

| Día | Demanda esperada |
|---|---|
| Lunes | 11,3 |
| Martes | 13,9 |
| Miércoles | 14,1 |
| Jueves | 13,2 |
| Viernes | 11,0 |
| Sábado | 5,7 |
| Domingo | 5,8 |
| *Demanda semanal* | *75* |

Para el pronóstico de los precios de la electricidad se consideran dos semanas típicas del mercado diario ($P1$ y $P2$), los cuales se exponen en la Tabla 5.3. Al mismo tiempo, con el fin de visualizar las fluctuaciones que presentan dichos pronósticos, estos se ilustran en la Figura 5.9.

A partir de los datos expuestos, se definen 6 instancias del caso de estudio (I.1-I.6) en función de los pronósticos detallados en la Figura 5.9 y de tres niveles de demanda esperada, las cuales se resumen en la Tabla 5.4. Es importante resaltar que, tanto los niveles de demanda como los pronósticos de precios de la energía utilizados, se especifican por hora sobre un horizonte semanal, $T = \{1,2,\ldots,168\}$ horas.

Por último, se debe tener en cuenta que la configuración de los equipos establece un tiempo de residencia mínimo y máximo de 3 y 8 horas, respectivamente.

Tabla 5.3. Pronóstico de los precios de energía para dos semanas ($/MWh).

| *P*1 | | | | | | | | *P*2 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hora | Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo | Hora | Lunes | Martes | Miércoles | Jueves | Viernes | Sábado | Domingo |
| 1 | 43,45 | 45,03 | 45,88 | 37,44 | 41 | 32,53 | 29,97 | 1 | 44,09 | 49,44 | 49,33 | 52,42 | 49,71 | 43,1 | 43,25 |
| 2 | 36,47 | 44,45 | 44,64 | 37,42 | 39,59 | 26,6 | 26,17 | 2 | 39,83 | 45,05 | 45,83 | 49,16 | 46,14 | 38,94 | 39,36 |
| 3 | 35,57 | 42,46 | 41,21 | 32,79 | 37,1 | 24 | 25,61 | 3 | 37,98 | 47,48 | 46,69 | 50,97 | 44,58 | 37,67 | 37,15 |
| 4 | 33,08 | 35,99 | 34,38 | 27,75 | 31,76 | 23,69 | 25,32 | 4 | 33,16 | 43,14 | 43,64 | 46,31 | 43,85 | 34,86 | 33,5 |
| 5 | 33,92 | 32,21 | 31,94 | 26,13 | 31,14 | 23,55 | 23,83 | 5 | 31,84 | 41,26 | 41,9 | 44,48 | 40,66 | 33,1 | 32,08 |
| 6 | 36,35 | 34,18 | 35,45 | 29,06 | 35,34 | 22,83 | 22,46 | 6 | 36,8 | 43,11 | 43,64 | 46,47 | 42,48 | 33,97 | 33,43 |
| 7 | 43,31 | 41,76 | 44,43 | 37,39 | 40,4 | 26,07 | 21,8 | 7 | 44,99 | 48,9 | 49,52 | 52,37 | 47,71 | 37,98 | 33,83 |
| 8 | 51,9 | 49,94 | 54,86 | 46,93 | 52,4 | 27,67 | 22,55 | 8 | 55,1 | 59,13 | 59,61 | 62,54 | 56,66 | 40,8 | 34,99 |
| 9 | 53,35 | 55,06 | 62,15 | 53,4 | 50,61 | 30,71 | 22,05 | 9 | 58,41 | 64,62 | 64,24 | 66,87 | 59,87 | 44,25 | 37,48 |
| 10 | 56,26 | 54,76 | 59,39 | 52,49 | 53,18 | 35,39 | 27,58 | 10 | 59,46 | 67,14 | 65,93 | 67,75 | 62,26 | 48,86 | 41,46 |
| 11 | 56,27 | 55,47 | 56,3 | 52,71 | 56,18 | 38,77 | 31,12 | 11 | 57,85 | 69,67 | 68,56 | 70,2 | 65,15 | 50,14 | 43,56 |
| 12 | 58,06 | 54,57 | 56,03 | 52,86 | 53,33 | 40,38 | 35,16 | 12 | 57,28 | 64,63 | 63,26 | 64,75 | 62,29 | 48,42 | 45,67 |
| 13 | 55,31 | 51,23 | 51,35 | 49,23 | 49,5 | 38,48 | 38,7 | 13 | 53,58 | 61,86 | 59,21 | 60,93 | 57,7 | 47,22 | 46,35 |
| 14 | 53,5 | 49,94 | 49,49 | 46,36 | 44,54 | 37,68 | 36,01 | 14 | 53,11 | 57,13 | 54,73 | 56,7 | 52,43 | 44,2 | 43,73 |
| 15 | 53,44 | 46,36 | 47,46 | 43,26 | 40,33 | 35,17 | 31,48 | 15 | 49,52 | 52,68 | 51,07 | 52,48 | 47,44 | 39,84 | 39,12 |
| 16 | 52,13 | 45,17 | 48,49 | 41,92 | 37,79 | 33,35 | 29,75 | 16 | 46,92 | 51,49 | 49,8 | 51,08 | 44,3 | 40,58 | 37,84 |
| 17 | 54,92 | 48,19 | 49,95 | 44,01 | 39,44 | 35,76 | 35,48 | 17 | 47,35 | 50,42 | 48,98 | 50,03 | 44,8 | 39,42 | 37,61 |
| 18 | 72,24 | 60,05 | 60,61 | 55,2 | 51,29 | 50,16 | 48,32 | 18 | 56,02 | 57,7 | 56,45 | 56,68 | 48,42 | 47,17 | 47,13 |
| 19 | 63,53 | 58,64 | 59,93 | 53,06 | 46,57 | 46,48 | 49,45 | 19 | 73,75 | 85,89 | 84,8 | 84,84 | 69,82 | 59,63 | 58,47 |
| 20 | 63,22 | 56,38 | 55,65 | 48,89 | 41,56 | 39,57 | 44,6 | 20 | 71,77 | 79,89 | 80,4 | 79,75 | 64,87 | 55,15 | 59,94 |
| 21 | 55,38 | 52,9 | 49,6 | 43,2 | 38,92 | 37,44 | 43,92 | 21 | 59,32 | 66,79 | 67,45 | 65,86 | 54,76 | 50,83 | 53,57 |
| 22 | 49,04 | 47,73 | 43,64 | 39,21 | 34,62 | 34,48 | 36,71 | 22 | 53,8 | 56,05 | 56,32 | 55,75 | 47,17 | 46,09 | 47,52 |
| 23 | 50,06 | 51,41 | 46,64 | 41,69 | 37,94 | 36,32 | 39,82 | 23 | 52,15 | 53,67 | 56,25 | 53,89 | 46,4 | 45,82 | 47,03 |
| 24 | 48,03 | 48,67 | 42,69 | 40,35 | 35,64 | 33,59 | 38,54 | 24 | 51,54 | 51,27 | 52,5 | 50,38 | 46,09 | 44,94 | 44,53 |

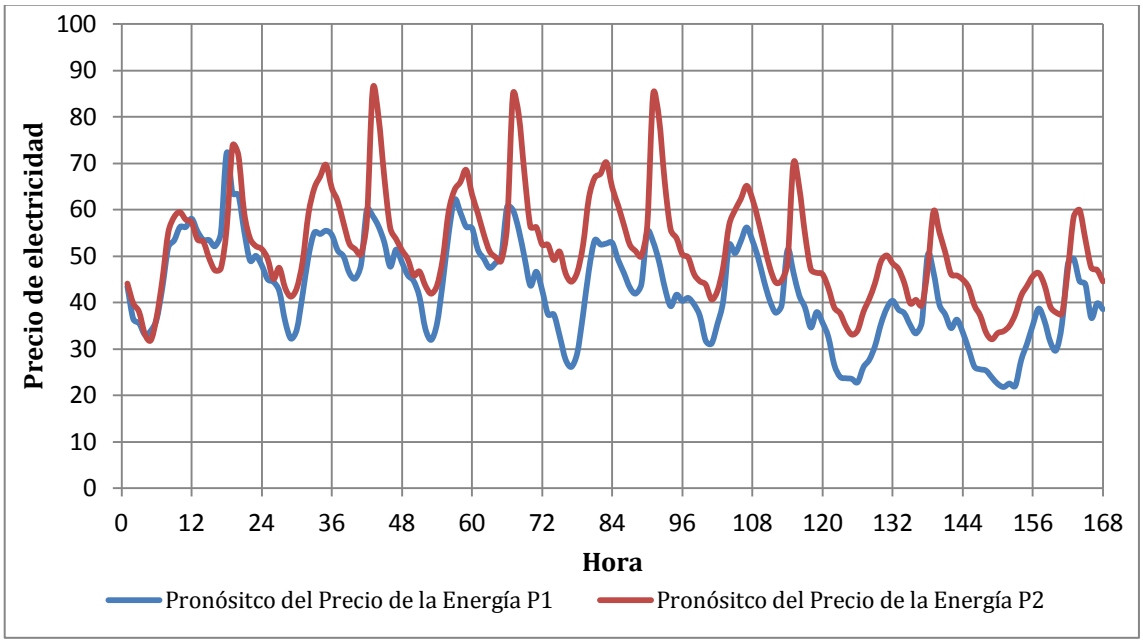



Figura 5.9. Pronósticos de precios de energía de dos semanas ($/MWh): $P1$ y $P2$.

Tabla 5.4. Definición de las instancias del caso de estudio abordado.

| Instancia del problema | Demanda esperada (unidad/día) | Pronóstico de precios de energía |
|---|---|---|
| I.1 | Demanda Esperada | P1 |
| I.2 | Demanda Esperada + 15% | P1 |
| I.3 | Demanda Esperada - 15% | P1 |
| I.4 | Demanda Esperada | P2 |
| I.5 | Demanda Esperada + 15% | P2 |
| I.6 | Demanda Esperada - 15% | P2 |

### 5.5.2. Comparación del modelo PSTN-MILP vs. modelo alternativo

Una vez definido el caso de estudio que se pretende evaluar, se resuelven mediante las formulaciones presentadas: (i) modelo PSTN-MILP y (ii) modelo alternativo (Mitra et al., 2012). El objetivo que se persigue es poder realizar un análisis comparativo de la

eficiencia computacional de ambos enfoques para las diferentes instancias detalladas del problema real correspondiente a una planta PSA con procesos de consumo intensivo de energía eléctrica.

Para realizar una adecuada comparación, los modelos se implementan en el mismo entorno computacional y bajo los mismos supuestos. Por esta razón, debido a que en Mitra et al. (2012) no se consideran restricciones de aceleración/desaceleración, estas no se tienen en cuenta en el análisis. En todos los casos, además, el cálculo de la función objetivo se realiza con la restricción (5.23), que computa el costo total de la energía necesaria para la operación de la planta.

Asimismo, es importante destacar que, inicialmente se toma como criterio de terminación el 0% de valor de optimalidad GAP y 3600 segundos de tiempo de CPU. Luego, a fin de realizar un análisis más detallado del modelo alternativo (Mitra et al., 2012), se consideran valores de GAP del 1% y 5%.

La Tabla 5.5 resume la información referida a las estadísticas computacionales de las 6 instancias detalladas en la Tabla 5.4, utilizando el optimizador CPLEX y tomando 0% de valor de GAP y 3600 segundos de CPU. Se puede observar que las soluciones para las instancias I.2-I.4 y I.6 son las mismas para ambos modelos. Sin embargo, el modelo alternativo no puede garantizar la optimalidad en el tiempo de CPU máximo predefinido.

Por otra parte, para los casos de I.1 e I.5, el modelo PSTN-MILP logra reportar la solución óptima en menos de 16,4 CPUs, mientras que el modelo alternativo puede proporcionar una solución factible con 2,52% y 1,74% de GAP, respectivamente, en 3600 CPUs. Por lo tanto, se puede observar claramente que, todas las instancias implementadas en el modelo PSTN-MILP, alcanzan la solución óptima en menos de 30 segundos de CPU, mientras que el modelo alternativo no puede garantizar la optimalidad en la mayoría de los casos, reportando sólo soluciones factibles en una 1 hora de tiempo computacional.

Tabla 5.5. Comparación del modelo PSTN-MILP con el modelo alternativo.

| Instancia | Modelo PSTN-MILP | | | | | | | Modelo alternativo (Mitra et al., 2012) | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Variables binarias | Variables continuas | Restricciones | Solución RMIP | Solución MIP | Valor GAP | Tiempo CPU (s) | Variables binarias | Variables continuas | Restricciones | Solución RMIP | Solución MIP | Valor GAP | Tiempo CPU (s) |
| I.1 | 2536 | 2858 | 9215 | 305,3 | 336,7 | 0 | 16,4 | 4207 | 1348 | 13531 | 299,1 | 337,2 | 2,52 | 3600 |
| I.2 | 2536 | 2858 | 9215 | 356,0 | 387,0 | 0 | 15,5 | 4207 | 1348 | 13531 | 355,8 | 387,0 | 1,81 | 3600 |
| I.3 | 2536 | 2858 | 9215 | 253,1 | 290,3 | 0 | 19,0 | 4207 | 1348 | 13531 | 250,7 | 290,3 | 2,60 | 3600 |
| I.4 | 2536 | 2858 | 9215 | 384,8 | 413,1 | 0 | 17,7 | 4207 | 1348 | 13531 | 381,3 | 413,1 | 0,49 | 3600 |
| I.5 | 2536 | 2858 | 9215 | 449,2 | 473,7 | 0 | 5,1 | 4207 | 1348 | 13531 | 447,1 | 474,2 | 1,74 | 3600 |
| I.6 | 2536 | 2858 | 9215 | 321,1 | 355,7 | 0 | 7,5 | 4207 | 1348 | 13531 | 316,6 | 355,7 | 0 | 2902 |

Tabla 5.6. Estadísticas computacionales del modelo alternativo con diferentes optimizadores y valores de GAP.

| Instancia | Valor de GAP=0,05 | | | | Valor de GAP=0,01 | | | | Valor de GAP=0 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CPLEX | | GUROBI | | CPLEX | | GUROBI | | CPLEX | | GUROBI | |
| | Solución MIP | Tiempo CPU (s) | Solución MIP | Tiempo CPU (s) | Solución MIP | Tiempo CPU (s) | Solución MIP | Tiempo CPU (s) | Solución MIP | Tiempo CPU (s) | Solución MIP | Tiempo CPU (s) |
| I.1 | 338,4 | 414,8 | 336,8 | 661,2 | 336,8 | 13432,7 | 336,7 | 2028,5 | 336,7 | 16883,4 | 336,7 | 3228,4 |
| I.2 | 387,6 | 345,7 | 387,0 | 148,4 | 387,0 | 6650,6 | 387,0 | 1745,7 | 387,0 | 13179,1 | 387,0 | 2407,7 |
| I.3 | 290,7 | 1990,7 | 290,3 | 672,0 | 290,6 | 7037,1 | 290,3 | 5202,9 | 290,3 | 10460,3 | 290,3 | 6965,3 |
| I.4 | 416,7 | 334,2 | 413,3 | 124,7 | 413,4 | 2257,1 | 413,1 | 922,5 | 413,1 | 3905,1 | 413,1 | 1327,2 |
| I.5 | 474,8 | 54,0 | 474,4 | 91,2 | 473,7 | 5547,7 | 473,7 | 901,7 | 473,7 | 8666,1 | 473,7 | 1743,2 |
| I.6 | 356,0 | 481,5 | 355,7 | 221,2 | 355,7 | 2178,7 | 355,7 | 976,7 | 355,7 | 2902,0 | 355,7 | 1152,3 |

Al mismo tiempo, se puede observar que los tamaños de los modelos difieren, aunque que el modelo alternativo utiliza menos variables continuas que el modelo PSTN-MILP, presenta más restricciones y prácticamente, el doble de variables binarias. Además, las soluciones relajadas de los modelos de programación mixta entera (RMIP), correspondientes al enfoque PSTN, se encuentran más cercanas de las soluciones MIP.

Como se comentó anteriormente, también se lleva a cabo un análisis del rendimiento computacional requerido por el enfoque alternativo (Mitra et al., 2012), utilizando el optimizador GUROBI y valores de GAP del 1% y 5%. Las soluciones halladas se muestran en la Tabla 5.6, donde se puede observar un mejor desempeño computacional en todos los casos donde se utiliza el optimizador GUROBI, en lugar de CPLEX. Tal es así, que se reporta una mejora en la solución de hasta un 0,8%, con respecto al optimizador CPLEX. Notar además que, no se considera un tiempo límite de terminación, por lo que en algunos casos tiempo computacional supera los 3600 segundo de tiempo de CPU impuestos en la primera comparación. A pesar del mejor desempeño computacional presentado por el modelo alternativo al utilizar GUROBI, este es ampliamente superado por el modelo PSTN-MILP propuesto.

### 5.5.3. Discusión de resultados

En esta subsección se presentan los resultados computacionales y el impacto económico para un conjunto de configuraciones operacionales implementadas en el modelo PSTN-MILP. En total se evalúan 6 escenarios, y al igual que antes, se considera una semana como horizonte de planificación. Es importante destacar que, se toma como base el caso de estudio sin variaciones en la demanda y con pronóstico $P1$, presentado en la sección 5.5.1, es decir, la instancia I.1 especificada en la Tabla 5.1. De este modo, se plantea como hipótesis que la demanda esperada y el pronóstico de los precios de energía son los mismos para todos los escenarios, variando sólo los tiempos permitidos de residencia (mínimos y máximos) en los modos globales. Además, se incluye el comportamiento del

aumento o disminución de la producción se tienen en cuenta las restricciones (5.15) - (5.17).

La Tabla 5.7 resume tanto, la información relativa a dichas configuraciones, con sus respectivos tiempos mínimos y máximos de permanencia en los modos operacionales globales (ON, OFF y ES), como las estadísticas computacionales que se obtienen para cada escenario (tamaño del modelo, tiempo computacional requerido y las soluciones reportadas). Notar que, en la última columna de la tabla, se reporta el número de veces que la planta detuvo el funcionamiento de sus equipos durante el horizonte analizado.

Tabla 5.7. Estadísticas computacionales implementando el modelo PSTN-MILP.

| Esc. | Tiempo mín. | Tiempo máx. | Var. bin. | Var. cont. | Restric. | Solución MIP | Valor GAP | Tiempo CPU (s) | Cantidad de Paradas |
|---|---|---|---|---|---|---|---|---|---|
| I.1.1 | 3 | 8 | 2536 | 2863 | 10727 | 348,4 | 0 | 33,57 | 9 |
| I.1.2 | 3 | 12 | 2536 | 2863 | 10715 | 318,8 | 0 | 4,76 | 7 |
| I.1.3 | 3 | 16 | 2536 | 2863 | 10703 | 308,7 | 0 | 2,51 | 6 |
| I.1.4 | 3 | 24 | 2536 | 2863 | 10679 | 306,4 | 0 | 3,42 | 6 |
| I.1.5 | 8 | 24 | 2536 | 2863 | 10679 | 318,4 | 0 | 13,35 | 5 |
| I.1.6 | 16 | 24 | 2536 | 2863 | 10679 | 338,4 | 0 | 8,14 | 4 |

Se puede observar que los casos I.1.3 y I.1.4 no muestran diferencias significativas, ni en las soluciones, ni en el rendimiento computacional. Esto puede deberse a que la restricción de permanencia máxima (con 16 y 24 horas, respectivamente) no se activa. Además, es importante mencionar que, al igual que en la comparación con el modelo alternativo, el modelo PSTN-MILP puede reportar las soluciones óptimas de todos los casos analizados, requiriendo un mínimo esfuerzo computacional. Más precisamente, en menos de 1 minuto de tiempo de CPU la formulación desarrollada puede garantizar la optimalidad, reportando las soluciones óptimas para todos los escenarios planteados.

A partir de los resultados obtenidos, se demuestra que el número de transiciones disminuye a medida que aumenta la flexibilidad de la configuración operativa, lo cual se refleja en términos de la cantidad de paradas de los equipos de la planta y en las soluciones MIP reportadas. Por ejemplo, el caso I.1.5 tiene mayores tiempos de residencia (tanto mínimo como máximo) que el caso I.1.1, lo que implica una menor cantidad de detenciones de la planta a lo largo del horizonte de planificación. Los programas de operaciones óptimos para ambos casos se muestran a través de diagramas Gantt en las Figuras 5.10 y 5.11, respectivamente.

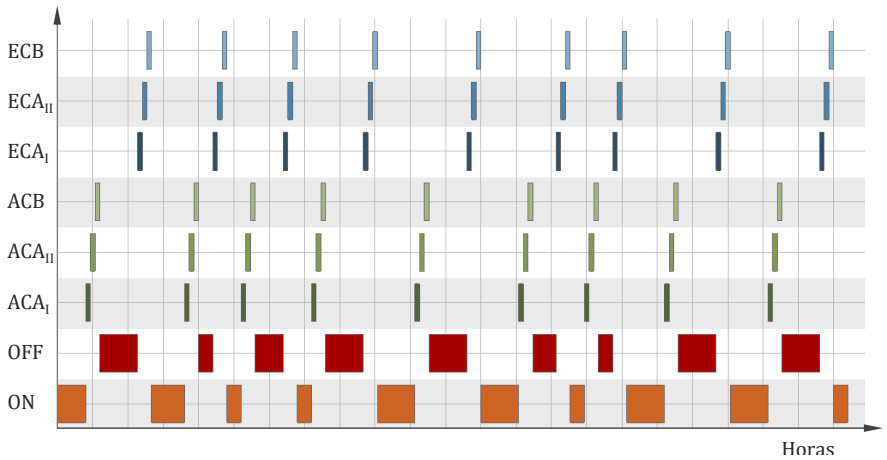



Figura 5.10. Programa óptimo de operaciones de producción para el escenario I.1.1.

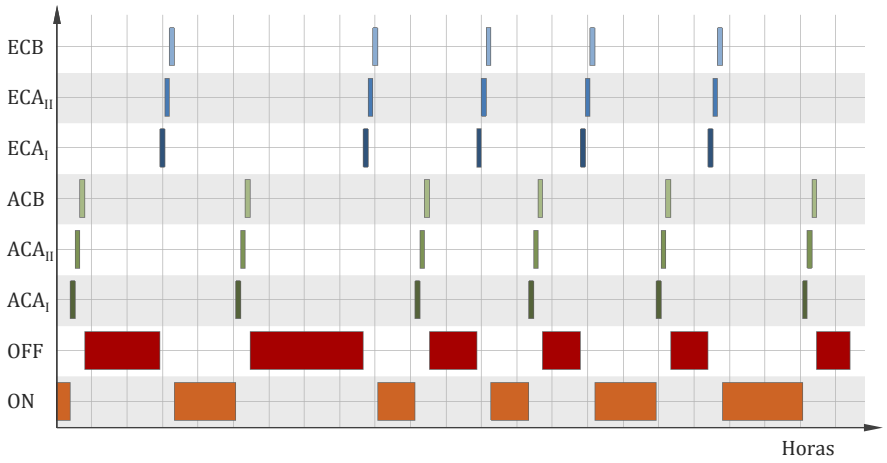



Figura 5.11. Programa óptimo de operaciones de producción para el escenario I.1.5.

Por otro lado, en función de las configuraciones permitidas, la demanda, el inventario y los precios de energía, puede que resulte más económico recurrir al estado ES durante pocas horas y luego, seguir produciendo (volver al estado ON), en lugar de apagar todo el equipamiento (OFF) y volver a encenderlo. Esta última opción, requiere más tiempo y probablemente, se incurra en mayores costos energéticos, ya que se debe cumplir con las transiciones asociadas a los procedimientos de apagado y encendido. Este es el caso del escenario I.1.2, donde la solución que optimiza los costos energéticos establece que se debe recurrir al estado "En espera" tres veces en la semana bajo estudio (ver Figura 5.12).

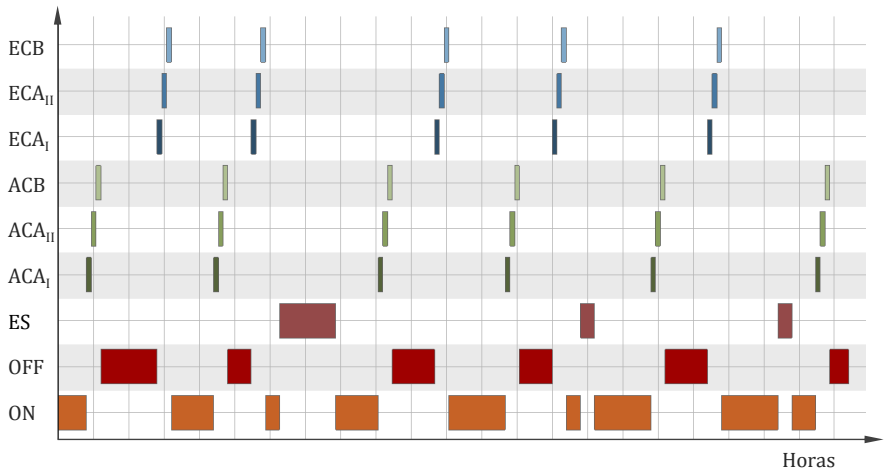



Figura 5.12. Programa óptimo de operaciones de producción para el escenario I.1.2.

A partir de los resultados obtenidos, se puede observar que para el mismo perfil de demanda, los costos totales de consumo de energía pueden diferir ya que al considerar diferentes configuraciones de equipos, el programa detallado óptimo en cuanto a los estados y consumos establecidos en cada período *t*. Asimismo, las mejoras que se alcanzan en los mencionados costos se deben a que la planta puede operar con una tasa máxima de producción durante los precios de energía más bajos, y apagar todos los equipos durante las horas de precios pico. Por lo tanto, las restricciones de tiempos mínimos y máximos de residencia deben ser lo suficientemente flexibles, para que configuración de la planta sea eficiente, en cuanto a períodos de producción y consumo de energía. La Figura 5.13 ilustra

este comportamiento a través de los resultados obtenidos para los caso I.1.1 y I.1.5, en términos de consumo de energía relativo y precios de electricidad ($P1$).

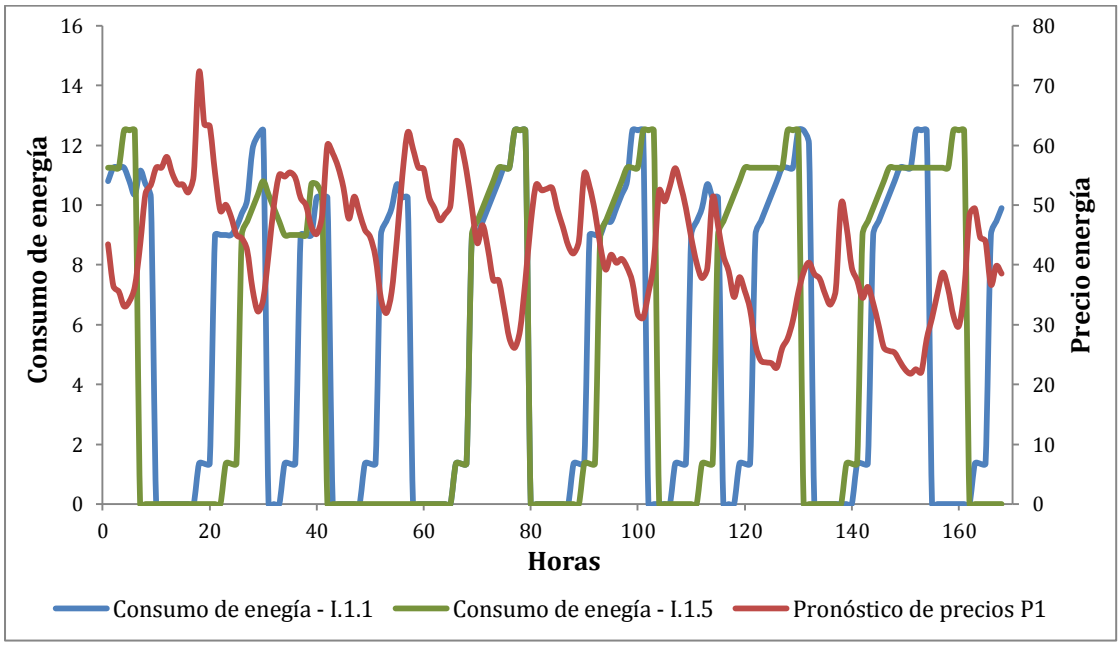



Figura 5.13. Perfiles de consumo de energía para los escenarios I.1.1 y I.1.5.

Se puede apreciar en la Figura 5.13 que, en ambos casos los niveles más altos de consumo de energía se dan durante las horas en las que el mercado diario ofrece precios bajos, lo cual se puede visualizar al final de la semana a partir de una leve tendencia decreciente de los pronósticos de precios. Al mismo tiempo, pero el caso contrario, se puede observar en el inicio de la semana, donde la planta detiene su producción debido a los altos precios de energía.

Además, notar que, debido a que el tiempo máximo de permanencia posible del escenario I.1.5, es mayor que el establecido para I.1.1, requiere menos cantidad de paradas de la planta, permitiéndole operar de manera continua en períodos de precios bajos, y por lo tanto, presentar un menor costo total de energía (Tabla 5.7).

Por último, los perfiles de inventario y niveles de producción correspondientes a los escenarios analizados (I.1.1 e I.1.5), se ilustran en la Figura 5.14. Se grafican los límites

inferiores y superiores de la capacidad de almacenamiento, de manera de verificar el cumplimiento de las restricciones de capacidad establecidas por el modelo. En la figura se observa que los perfiles de inventario de ambos escenarios presentan curvas equivalentes (con aumento de niveles de inventario durante las horas de normal funcionamiento de la planta, y disminución en las horas en las que no hay producción, donde la demanda se satisface a partir de los productos almacenados), pero desfasadas debido a las diferentes restricciones de configuración.

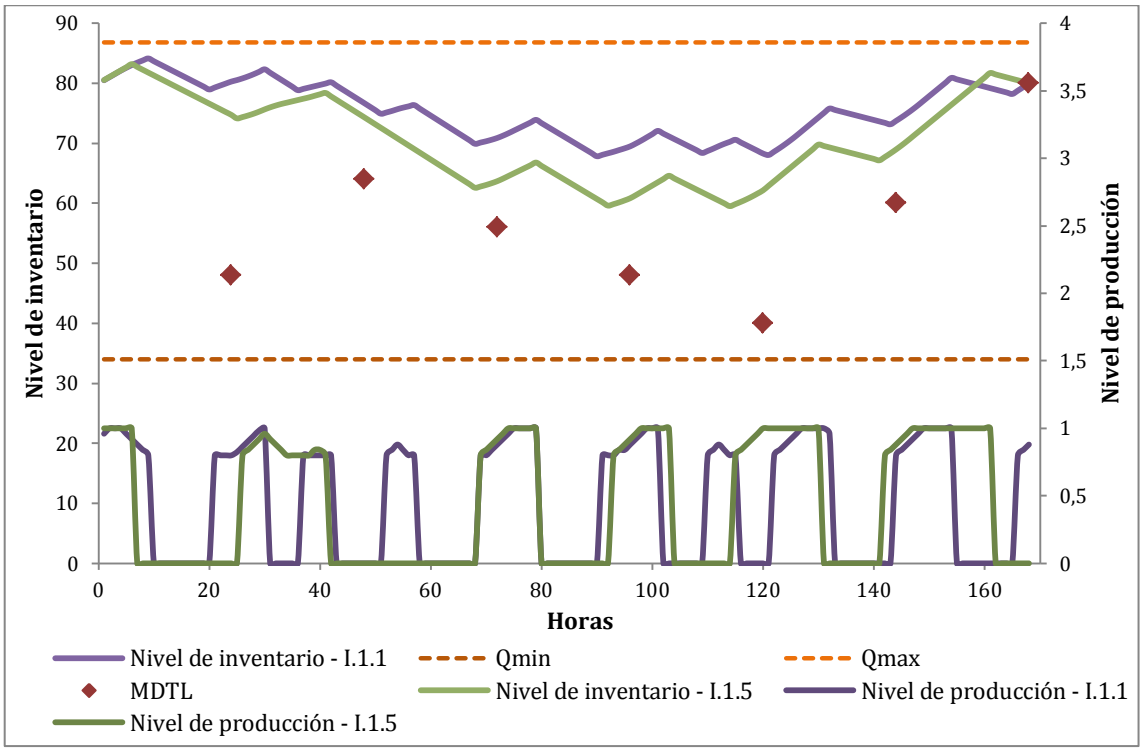



Figura 5.14. Perfiles de inventario y producción para los escenarios I.1.1 y I.1.5.

Notar que, el escenario I.1.1 presenta mayores niveles de inventario promedio a lo largo del horizonte de planificación. Se satisfacen los niveles mínimos de inventario, tanto de forma horaria, como al final del horizonte de planificación. Esto último le permite a la empresa poder empezar la siguiente semana de trabajo con un mínimo nivel de producto almacenado.

## 5.6. Conclusiones

En este capítulo se ha abordado el problema de programación de las operaciones de plantas que presentan un consumo intensivo de la energía eléctrica (CIEE) y participan en mercados de energía donde los precios varían continuamente, en particular en el marcado "Day-ahead".

La estrategia de solución presentada está basada en un nuevo concepto de representación sistemática de los modos operativos de las plantas CIEE, llamado PSTN, y en una formulación MILP de tiempo discreto (PSTN-MILP), donde el principal objetivo es evaluar las decisiones reactivas que pueda tener la planta bajo estudio, en función de los cambios en los precios de la energía, de manera de minimizar el costo total asociado al consumo de la misma.

La nueva formulación PSTN-MILP permite representar, de manera uniforme, los estados operativos y modelar el comportamiento dinámico de este tipo de procesos, a través de transiciones de estados. La principal ventaja del enfoque PSTN es que da lugar a una formulación de programación de tiempo discreto que es ligeramente más estricta, y computacionalmente superior, a los modelos MILP presentados en la literatura hasta el momento.

La estrategia de solución fue implementada en un caso de estudio de una planta industrial de separación de aire. Se realizó una comparación entre el modelo PSTN-MILP y la formulación presentada previamente por Mitra et al. (2012), con el fin de evaluar la eficiencia computacional a partir de la resolución de diferentes instancias del problema (variando los niveles de demandas esperadas y los pronósticos de precios de energía). Los resultados establecieron que, a pesar del gran tamaño que puede llegar a tener el modelo PSTN-MILP, tiene la capacidad de optimizar las decisiones operativas de la planta teniendo en cuenta las fluctuaciones de los precios y reportar la solución óptima en pocos segundos de tiempo de CPU. De este modo, desde una perspectiva computacional, el modelo mostró soluciones con 0 % de GAP en muy buenos tiempos de cómputo (menos de un minuto de

tiempo de CPU). Además, a partir del análisis realizado se demostró que el modelo propuesto supera considerablemente al de la literatura (Mitra et al., 2012). Por lo tanto, se debe tener en cuenta que las formulaciones alternativas MILP de tiempo discreto, basadas en diferentes criterios, pueden tener una amplia diferencia en términos de rendimiento computacional y calidad de la solución.

Por lo expuesto, se puede concluir que el nuevo modelo matemático PSTN-MILP es computacionalmente muy eficiente y robusto, capaz de resolver problemas industriales de programación de las operaciones de plantas CIEE. Asimismo, la formulación propuesta puede ser fácilmente adaptada a otras configuraciones operacionales para encontrar el programa óptimo de operaciones en un tiempo de cómputo aceptable, y utilizada como herramienta útil para para la aplicación a plantas industriales de elevado consumo de energía.

## *Nomenclatura*

### *Conjuntos*

| | |
|---|---|
| $T$ | conjunto de períodos de tiempo (índice $t$) |
| $M$ | conjunto de modos de operación (índice $m$) |
| $S$ | conjunto de estados operativos (índice $s$) |
| $D$ | conjunto de días del horizonte de planificación (índice $d$) |
| $T^{last}$ | subconjunto de períodos de tiempo de finalización de cada día |
| $S^{initial}$ | subconjunto de estados secuenciales iniciales de los modos ON, OFF y ES |
| $S^{inter}$ | subconjunto de estados de transición intermedios de los modos ON, OFF y ES |

$S^{critical}$ subconjunto de estados críticos de transición de los modos ON, OFF y ES

$S^{down-initial}$ subconjunto de estados iniciales del proceso de apagado de la planta

$S^{up-initial}$ subconjunto de estados iniciales del proceso de encendido de la planta

$S^{down-inter}$ subconjunto de estados intermedios del proceso de apagado de la planta

$S^{up-inter}$ subconjunto de estados intermedios del proceso de encendido de la planta

$S^{minProd}$ subconjunto de estados con producción mínima

$LIC_s$ subconjunto de estados que preceden inmediatamente a un estado crítico $s$

$NTS_s$ subconjunto de estados que pueden suceder inmediatamente al estado crítico $s$

***Parámetros***

$MinP_s$ nivel producción mínima permitida por hora en el estado $s$

$MaxP_s$ nivel producción máxima permitida por hora en el estado $s$

$MDTL_{T^{last}}$ niveles mínimos de almacenamiento al final del día

$ED_t$ demanda esperada en el período $t$

$FPC_s$ consumo fijo de energía eléctrica

$VPC_s$ consumo variable de energía eléctrica

$EP_t$ pronóstico del precio de la energía en el período $t$

$Qmin$ nivel mínimo de almacenamiento

$Qmax$ nivel máximo de almacenamiento

| | |
|---|---|
| $EP^{FIXED}$ | precio fijo promedio de energía de una semana |
| $I_0$ | nivel inicial de inventario |
| $T^{last}$ | tiempo final del horizonte de planificación |
| $nd$ | número de estados intermedios en el proceso de apagado de la planta |
| $ns$ | número de estados intermedios en el proceso de encendido de la planta |
| $min_{res}$ | tiempo mínimo de permanencia de los estados ON, OFF y ES |
| $max_{res}$ | tiempo máximo de permanencia de los estados ON, OFF y ES |
| $mn$ | tiempo mínimo de permanencia en estados críticos de transición |
| $mx$ | tiempo máximo de permanencia en estados críticos de transición |
| $inc$ | porcentaje permitido de aumento de la producción en el proceso de encendido de la planta |
| $dec$ | porcentaje permitido de disminución de la producción en el proceso de apagado de la planta |

***Variables Continuas***

| | |
|---|---|
| $P_{s,t}$ | nivel de producción del estado $s$ en el período $t$ |
| $PW_t$ | consumo de energía en el período $t$ |
| $I_t$ | inventario disponible al final del período $t$ |
| $Cost$ | costo total de energía eléctrica |

***Variables binarias***

| | |
|---|---|
| $W_{s,t}$ | indica si la planta opera en el estado $s$ durante el período de tiempo $t$ |

# Capítulo 6

# *Conclusiones finales y trabajos futuros*

## 6.1. Conclusiones finales

En la actualidad, las industrias están inmersas en mercados globales altamente competitivos, los cuales se caracterizan por su flexibilidad, rapidez en sus respuestas y por sus indicadores de performance, tales como satisfacción al cliente, costos de manufactura y de inventarios, tiempos de entrega, entre otros. Teniendo en cuenta dichos requerimientos surge la necesidad de programar eficientemente la producción con el fin de mejorar la operatividad de los ambientes productivos y coordinar el uso de los recursos, reduciendo tiempos ociosos, evitando niveles elevados de inventario, etc. De este modo, en el contexto de un proceso complejo de toma de decisiones, resulta imprescindible para las industrias la optimización de las operaciones productivas, de manera de generar un gran impacto en la productividad de todo el proceso de fabricación.

Frente a este panorama, la presente tesis ha abordado el desarrollo de distintas técnicas, procedimientos sistemáticos y algoritmos híbridos basados en formulaciones matemáticas y reglas heurísticas, como respuesta a las necesidades de las compañías de gestionar sus procesos operativos en forma eficiente. En particular, se propusieron diferentes estrategias de solución para resolver problemas de programación de operaciones de corto y mediano plazo en diversos ambientes industriales.

Tal como se ha mencionado en el primer capítulo de esta tesis doctoral, el objetivo principal de la misma ha sido desarrollar diferentes herramientas de soporte a los procesos de toma de decisiones para la gestión eficiente de las operaciones de producción presentes en sistemas de manufactura de diversos entornos industriales. De este modo, se han propuestos estrategias basadas en programación matemática lineal mixta-entera (MILP), técnicas heurísticas de descomposición-agregación y procedimientos iterativos, para la optimización de diferentes problemas de "scheduling". A través de dichas herramientas, que han sido presentadas a lo largo de este trabajo, se intentó superar ciertas limitaciones y realizar aportes innovadores respecto de los enfoques publicados en la literatura del área bajo estudio.

En primer lugar, se introdujeron enfoques exactos basados en tres formulaciones matemáticas alternativas MILP, con dominio continuo del tiempo, para la resolución del problema de programación de las operaciones presente en la industria naval. Dichas formulaciones están basadas en los conceptos de: (i) "time-slots", (ii) precedencia inmediata y (iii) precedencia general. En particular, el problema abordado considera el proceso de fabricación y montaje de grandes embarcaciones con fines militares y comerciales, donde se debe considerar la manipulación de grandes productos intermedios, la coordinación de un importante número de recursos y además, operaciones de ensamblado que se llevan a cabo en varias etapas de la secuencia de procesamiento. Desde el punto de vista operacional, el proceso de manufactura se considera un sistema productivo multietapa multiproducto con unidades idénticas en paralelo, es decir, un flow shop flexible, que tiene la particularidad de tener etapas destinadas a actividades de

ensamblado de productos intermedios. Se presentó un caso de estudio real de tamaño industrial, correspondiente a un astillero europeo, dedicado a la construcción de diversas embarcaciones. A partir de dicho caso de estudio se definieron diferentes instancias para evaluar el desempeño de los métodos propuestos para su resolución, teniendo en cuenta, tanto la calidad de las soluciones obtenidas, como del tiempo computacional requerido. A partir de los resultados obtenidos, se puede concluir que:

- Sin bien los métodos exactos presentados no son capaces de reportar la solución óptima en un tiempo razonable para problemas de tamaño real, el enfoque de precedencia general fue el que presentó el mejor desempeño computacional de los 3 desarrollados.

- Los problemas de programación de escala industrial resultan ser irresoluble de manera óptima en un tiempo de cálculo razonable, debido a su complejidad combinatoria, que involucra miles de variables y restricciones asociadas a múltiples productos intermedios y etapas de procesamiento.

- Las herramientas de programación matemática comúnmente utilizadas en el área de ingeniería de procesos, como las formulaciones matemáticas rigurosas monolíticas en este caso, no son capaces de reportar soluciones prácticas a problemas de programación detallada.

A continuación, con el objetivo de dar solución a problemas industriales de "scheduling" de gran tamaño, como el antes mencionado, se propuso una metodología superadora a la anterior, la cual está basada en un procedimiento iterativo de construcción y mejora y en un modelo matemático MILP. Este algoritmo, primero resuelve modelos simplificados que componen una primera etapa, llamada "etapa de construcción", de la cual se obtiene una solución inicial factible. Luego, en una segunda etapa ("etapa de mejora"), dicha solución se va mejorando de manera iterativa, hasta obtener una solución muy eficiente con un tiempo computacional aceptable, desde el punto de vista industrial. Esta estrategia fue aplicada a dos casos de estudios reales, uno de los cuales corresponde

al caso de estudio antes mencionado. Los resultados obtenidos fueron comparados con aquellos presentados por la formulación MILP de precedencia generalizada presentada anteriormente, y se pudo observar que:

- El algoritmo pudo reportar un programa de operaciones factible y de alta calidad (óptimo o cuasi-óptimo), en todos los casos evaluados, en tan sólo unos pocos segundos de tiempo de CPU.
- La estrategia de solución propuesta presenta una mayor eficiencia que los métodos exactos, para problemas de tamaño mediano y grande.
- Resulta muy conveniente descomponer el problema de "scheduling" de sistemas productivos de escala industrial en modelos MILP de tamaño reducido y resolverlo iterativamente, en lugar de recurrir a un modelo MILP monolítico.
- También se debe destacar que la aplicación de la herramienta computacional a los dos casos de estudio, permitió obtener una mejora en el makespan del 15,2% y 14%, respectivamente, requiriendo un mínimo esfuerzo computacional, con respecto al programa de operaciones reportado por el método exacto.

Posteriormente, se propuso una nueva formulación MILP sustentada en una representación continua del tiempo y en el concepto de precedencia generalizada para sistemas de manufactura flexible, con el fin de llevar a cabo decisiones en cuanto a: (i) la asignación de un conjunto de tareas a unidades, las cuales pueden realizar operaciones de más de una etapa del proceso productivo, (ii) la secuenciación de dichas tareas, las cuales pueden corresponder a distintas etapas, en unidades multipropósito y (iii) el rediseño de la configuración del sistema bajo estudio. Asimismo, un nuevo enfoque algorítmico iterativo, basado en modelos de programación matemática MILP, ha sido introducido para abordar de manera integrada la generación del programa de operaciones y el rediseño de diversas plantas multiproducto multipropósito, denominadas job shop flexible. De manera similar a la técnica mencionada anteriormente, pero con mayor flexibilidad, se genera un

programa de operaciones inicial que se va mejorando de manera iterativa. Asimismo, en este caso, en la última etapa iterativa del algoritmo se propone el análisis de posibles rediseños factibles de la planta de manufactura, que permitan lograr una mayor eficiencia en el uso de las unidades de procesamiento multipropósito, y al mismo tiempo, mejorando el valor de la función objetivo. Esta metodología pudo ser extendida también a sistemas de manufacturas con operaciones de ensamblado de productos intermedios. Entre los principales logros alcanzados se destacan:

- La posibilidad de incorporar reglas heurísticas y criterios de preordenamiento a la formulación matemática propuesta, con el fin de desarrollar modelos reducidos y considerar decisiones, tanto de programación como de rediseño, que permiten resolver problemas reales presentes en diferentes entornos industriales.

- La nueva propuesta fue aplicada a diversas instancias de casos de estudio de gran escala de la industria automotriz y naval.

- A partir de los resultados obtenidos se demostró la eficiencia de la estrategia de descomposición de programación y rediseño, tanto en el tiempo de CPU requerido para la resolución, como en la calidad de las soluciones obtenidas.

- Además, se debe destacar la fácil adaptabilidad de la metodología propuesta a diferentes problemas de programación de operaciones.

Otro aporte de esta tesis fue el desarrollo de un modelo MILP con representación discreta del tiempo, para optimizar el programa de operaciones de plantas que, para su normal operación, requieren energía eléctrica en grandes cantidades y por lo tanto, sus costos operativos dependen en gran medida de la sensibilidad a las fluctuaciones de los precios de la energía en el tiempo. Específicamente, para la resolución del problema de "scheduling" la formulación tiene en cuenta: (i) pronósticos de precios de energía eléctrica para un horizonte de tiempo, (ii) demanda esperada y (iii) restricciones operativas, tales

como modos o estados de operación y de transición factibles, tiempos de residencia mínimos y máximos, niveles de inventario permitidos y tasas de producción mínimas y máximas, basadas en un esquema de estados de la planta. Teniendo como objetivo principal encontrar un programa predictivo de operaciones de producción óptimo durante un horizonte de tiempo determinado, que satisfaga la demanda y minimice el costo total de energía consumida, se diseñó además, una representación sistemática para el proceso de transición de estados de operación de la planta, llamada "Process State Transition Network" (PSTN). Los principales avances obtenidos se resumen a continuación:

- La potencialidad de la formulación matemática MILP desarrollado fue testeada a través de la resolución de distintos escenarios derivados de un mismo caso de estudio del mundo real, correspondiente a una planta de separación de aire.

- Los resultados demostraron que, a pesar del gran tamaño del modelo MILP en cuanto a restricciones y variables binarias y continuas, este es capaz de reportar la solución óptima para todos los casos.

- Específicamente, se pudo observar que el modelo de optimización pudo obtener la solución óptima para todos los casos analizados en menos de un minuto de tiempo de CPU, superando los enfoques reportados en la literatura.

- La principal ventaja de la representación PSTN es que da lugar a una formulación de programación matemática de tiempo discreto que es ligeramente más estricta y computacionalmente superior a los modelos MILP publicados en el área.

- El nuevo modelo PSTN-MILP se puede adaptar fácilmente a otras configuraciones operativas para encontrar el programa óptimo de operaciones en un tiempo de cálculo razonable. Asimismo, podría extenderse

fácilmente a la reprogramación de entornos de producción extremadamente dinámicos.

Debido a la alta complejidad expuesta para los problemas de producción abordados en esta tesis, los cuales están basados en diferentes configuraciones de sistemas de manufactura, surgió la necesidad imprescindible de desarrollar enfoques o estrategias de solución para cada caso. Asimismo, se debe destacar que, las herramientas computacionales fueron desarrolladas con el objetivo de reportar buenas soluciones en tiempos de cómputo razonables.

Por otra parte, se debe considerar que los procedimientos implementados conducen a una marcada reducción en los tamaños de los modelos, lograda fundamentalmente a través del desacople de las decisiones de asignación y secuenciación mediante un manejo de diferentes conjuntos de variables binarias y por la implementación de reglas heurísticas y modelos matemáticos con tamaños reducidos. De este modo, en todos los casos se logró mejorar en forma significativa los programas de operaciones en piso de planta reportados por otros autores para los mismos ejemplos, tanto desde el punto de vista de la calidad de las soluciones halladas (makespan y costos totales, según corresponda), como en la eficiencia computacional.

Finalmente, se debe destacar que, en todos los casos de estudio analizados en esta tesis, los resultados obtenidos fueron altamente satisfactorios. Tal es así que, como se ha expresado al inicio de este trabajo de tesis, algunos de los desarrollos alcanzados han sido reportados en diversas revistas científicas y congresos internacionales, y los restantes, enviados para su publicación en revistas referentes del el área de investigación de la tesis.

## 6.2. Trabajos futuros

Las estrategias y herramientas avanzadas de solución desarrolladas en esta tesis involucran, en particular, la programación detallada de operaciones de diferentes tipos de plantas productivas. De este modo, se considera la gestión de las operaciones de

producción como soporte para los procesos de toma de decisiones referentes a las actividades de manufactura de procesos industriales complejos de gran escala.

Por esta razón, se plantean como trabajos futuros la continuación de las líneas de investigación abordadas a lo largo de este trabajo de tesis: (i) la programación de las operaciones de sistemas de manufactura flexibles para el proceso de construcción de grandes embarcaciones inherente a la industria naval, y por otro lado, (ii) la programación detallada de las operaciones en procesos productivos con un intensivo consumo de energía eléctrica y que participan en diversos mercados de energía.

En el primer caso, se propone como trabajo futuro adaptar las estrategias de solución basadas en formulaciones matemáticas y reglas heurísticas de descomposición-agregación propuestas en esta tesis, para considerar la programación de las operaciones de montaje de una flota compuesta por un número determinado de embarcaciones y evaluar la inversión de un dique seco adicional al original, de manera de aumentar la flexibilidad y productividad, y por lo tanto, los beneficios de la compañía.

Por otra parte, se propone extender la formulación MILP determinista presentada en el Capítulo 5 para considerar la producción de múltiples productos, e incorporar las ideas principales del modelo PSTN a un nuevo marco basado en la programación estocástica, con el fin de abordar la incertidumbre en los datos de los precios de la energía y su repercusión en la toma de decisiones. De esta manera, se considerará operar de manera dinámica, tanto en el mercado del día anterior, como en el mercado spot.

Asimismo, se propone el desarrollo de un algoritmo para formular un modelo reactivo y de este modo, proveer una herramienta capaz de revisar y corregir el "schedule", luego de la ocurrencia de eventos imprevistos, tales como las fluctuaciones en los precios de energía y la actualización diaria de los pronósticos de dichos precios. Por consiguiente, la formulación matemática originalmente propuesta será extendida para proporcionar una herramienta sistemática capaz de actualizar diariamente la información y el programa de operaciones, es decir, se trabajará con un horizonte de tiempo rodante.

Se analizará, de este modo, el problema de reprogramación de operaciones en ambientes de producción dinámico, como el de las plantas de separación de aire estudiado en la tesis.

Por último, se propone abordar una nueva línea de investigación, relativa a mejorar la gestión y operación de los sistemas de producción, almacenamiento y transporte a través del desarrollo de estrategias de solución basadas en modelos matemáticos, coordinando e integrando estrategias de recolección con procesos de producción e inventariado y estrategias de distribución, tratando de implementar prácticas logísticas avanzadas. En particular, los esfuerzos futuros de investigación se concentrarán en introducir nuevas herramientas avanzadas de solución para la distribución de bienes, en el contexto de cadenas integradas de producción y distribución, a través de la creación de técnicas híbridas y sistemáticas para la gestión de operaciones en redes de distribución de múltiples niveles.

# *Referencias bibliográficas*


Aguirre, A., Müller, E., Seffino, S., Méndez, C.A., 2008. Applying a simulation-based tool to productivity management in an automotive-parts industry. Proc. - Winter Simul. Conf. 1838–1846.

Aguirre, A.M., Méndez, C.A., Gutierrez, G., De Prada, C., 2012. An improvement-based MILP optimization approach to complex AWS scheduling. Comput. Chem. Eng. 47, 217–226.

Artigues, C., Lopez, P., Haït, A., 2013. The energy scheduling problem: Industrial case-study and constraint propagation techniques. Int. J. Prod. Econ. 143, 13–23.

Baldea, M., Harjunkoski, I., 2014. Integrated production scheduling and process control: A systematic review. Comput. Chem. Eng. 71, 377–390.

Banks, J., Carson, J.S., Nelson, B.L., Nicol, D.M., 2010. Discrete-Event System Simulation, FIFTH EDIT. ed, Prentice-Hall. Pearson.

Basán, N.P., Achkar, V.G., Garcia-del-valle, A., Méndez, C.A., 2017a. An effective continuous-time formulation for scheduling optimization in a shipbuilding assembly process. 29th Eur. Model. Simul. Symp. EMSS 2017 83–90.

Basán, N.P., Achkar, V.G., Mendez, C.A., Garcia-del-Valle, A., 2017b. A heuristic simulation-based framework to improve the scheduling of blocks assembly and the production process in shipbuilding. 2017 Winter Simulation Conference (WSC). IEEE, pp. 3218–3229.

Basán, N.P., Achkar, V.G., Mendez, C.A., Garcia-Del-Valle, A., 2017c. A Hybrid Simulation-Based Optimization Approach for Scheduling, in: Proceedings of XLIII CLEI - 46 JAIIO. pp. 57–68.

Basán, N.P., Achkar, V.G., Méndez, C.A., Garcia-del-valle, A., 2017d. A heuristic simulation-based framework to improve the scheduling of blocks assembly and the production process in shipbuilding. Winter Simul. Conf. WSC 2017 F134102, 3218–3229.

Basán, N.P., Coccola, M.E., Mendez, C.A., 2014. Optimizing the design and operation of a beer

packaging line through an advanced simio-based DES tool, in: Proceedings of the Winter Simulation Conference 2014. IEEE, pp. 1989–2000.

Basán, N.P., Cóccola, M.E., Méndez, C.A., 2015. Conducting experimental design and optimization on an innovative car rental business. 27th Eur. Model. Simul. Symp. EMSS 2015 166–171.

Basán, N.P., Pulido, R., Cóccola, M., Méndez, C.A., 2017e. Aerospace Manufacturing Industry: A Simulation-based Decision Support Framework for the Scheduling of Complex Hoist Lines. Iberoam. J. Ind. Eng. 9, 100–117.

Benttaleb, M., Hnaien, F., Yalaoui, F., 2018. Two-machine job shop problem under availability constraints on one machine: Makespan minimization. Comput. Ind. Eng. 117, 138–151.

Blazewicz, J., Dror, M., Weglarz, J., 1991. Mathematical programming formulations for machine scheduling: A survey. Eur. J. Oper. Res. 51, 283–300.

Brandimarte, P., Torino, P., 1993. Routing and scheduling in a flexible job shop by tabu search. Ann. Oper. Res. 41, 157–183.

Brucker, P., Schlie, R., 1990. Job-shop scheduling with multi-purpose machines. Computing 45, 369–375.

Castro, P.M., Aguirre, A.M., Zeballos, L.J., Méndez, C.A., 2011a. Hybrid Mathematical Programming Discrete-Event Simulation Approach for Large-Scale Scheduling Problems. Ind. Eng. Chem. Res. 50, 10665–10680.

Castro, P.M., Barbosa-Póvoa, A.P., Novais, A.Q., 2005. A design and scheduling RTN continuous-time formulation. Comput. Aided Chem. Eng. 20, 1213–1218.

Castro, P.M., Grossmann, I.E., 2006. An efficient MILP model for the short-term scheduling of single stage batch plants. Comput. Chem. Eng. 30, 1003–1018.

Castro, P.M., Grossmann, I.E., 2005. New Continuous-Time MILP Model for the Short-Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. 44, 9175–9190.

Castro, P.M., Grossmann, I.E., Zhang, Q., 2018. Expanding scope and computational challenges in process scheduling. Comput. Chem. Eng. 114, 14–42.

Castro, P.M., Harjunkoski, I., Grossmann, E., 2010. Effective Decomposition Algorithm for Multistage Batch Plant Scheduling, Computer Aided Chemical Engineering. Elsevier B.V.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2011b. Optimal scheduling of continuous plants with energy constraints. Comput. Chem. Eng. 35, 372–387.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2009. Optimal Short-Term Scheduling of Large-Scale Multistage Batch Plants. Ind. Eng. Chem. Res. 48, 11002–11016.

Castro, P.M., Harjunkoski, I., Grossmann, I.E., 2009. New Continuous-Time Scheduling Formulation for Continuous Plants under Variable Electricity Cost. Ind. Eng. Chem. Res. 48, 6701–6714.

Castro, P.M., Zeballos, L.J., Méndez, C.A., 2012. Hybrid time slots sequencing model for a class of scheduling problems. AIChE J. 58, 789–800.

Cebral-Fernandez, M., Crespo-Pereira, D., Garcia-Del-Valle, A., Rouco-Couzo, M., 2016. Improving planning and resource utilization of a shipbuilding process based on simulation, in: 28th European Modeling and Simulation Symposium, EMSS 2016. pp. 197–203.

Cerdá, J., Henning, G.P., Grossmann, I.E., 1997. A Mixed-Integer Linear Programming Model for Short-Term Scheduling of Single-Stage Multiproduct Batch Plants with Parallel Lines. Ind. Eng. Chem. Res. 36, 1695–1707.

Chen, X., Chau, V., Xie, P., Sterna, M., Błażewicz, J., 2017. Complexity of late work minimization in flow shop systems and a particle swarm optimization algorithm for learning effect. Comput. Ind. Eng. 111, 176–182.

Cheng, C.-Y., Ying, K.-C., Chen, H.-H., Lin, J.-X., 2018. Optimization algorithms for proportionate flowshop scheduling problems with variable maintenance activities. Comput. Ind. Eng. 117, 164–170.

Chibeles-Martins, N., Pinto-Varela, T., Barbósa-Póvoa, A.P., Novais, A.Q., 2010. A meta-heuristics approach for the design and scheduling of multipurpose batch plants, Computer Aided Chemical Engineering. Elsevier B.V.

Cho, K.K., Oh, S.J., Ryu, K.R., Choi, H.R., 1998. An Integrated Process Planning and Scheduling System for Block Assembly in Shipbuilding. CIRP Ann. - Manuf. Technol. 47, 419–422.

Choi, S.H., Wang, K., 2012. Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach. Comput. Ind. Eng. 63, 362–373.

Chu, Y., You, F., Wassick, J.M., 2014. Hybrid method integrating agent-based modeling and heuristic tree search for scheduling of complex batch processes. Comput. Chem. Eng. 60, 277–296.

Ciccantell, P.S., Shin, K., 2009. The Steel And Shipbuilding Industries Of South Korea: Rising East Asia And Globalization. Climate Change 2013 - The Physical Science Basis. Cambridge University Press, Cambridge, pp. 167–192.

Cóccola, M.E., Dondo, R., Méndez, C.A., 2015. A MILP-based column generation strategy for managing large-scale maritime distribution problems. Comput. Chem. Eng. 72, 350–362.

Corsano, G., Montagna, J.M., Iribarren, O.A., Aguirre, P.A., 2007. Heuristic method for the optimal synthesis and design of batch plants considering mixed product campaigns. Ind. Eng. Chem.

Dauzère-Pérès, S., Paulli, J., 1997. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann. Oper. Res. 70, 281–306.

Demir, Y., Kürşat Işleyen, S., 2013. Evaluation of mathematical models for flexible job-shop scheduling problems. Appl. Math. Model. 37, 977–988.

Eberle, L., Capón-García, E., Sugiyama, H., Graser, A., Schmidt, R., Hungerbühler, K., 2016. Rigorous approach to scheduling of sterile drug product manufacturing. Comput. Chem. Eng. 94, 221–234.

Fattahi, P., Jolai, F., Arkat, J., 2009. Flexible job shop scheduling with overlapping in operations. Appl. Math. Model. 33, 3076–3087.

Fattahi, P., Saidi Mehrabad, M., Jolai, F., 2007. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. J. Intell. Manuf. 18, 331–342.

França, P.M., Gendreau, M., Laporte, G., Müller, F.M., 1996. A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. Int. J. Prod. Econ. 43, 79–89.

Fumero, Y., Corsano, G., Montagna, J.M., 2012a. Planning and scheduling of multistage multiproduct

batch plants operating under production campaigns. Ann. Oper. Res. 199, 249–268.

Fumero, Y., Corsano, G., Montagna, J.M., 2012b. A Mixed Integer Linear Programming model for simultaneous design and scheduling of flowshop plants. Appl. Math. Model. 37, 1652–1664.

Garey, M.R., Johnson, D.S., Sethi, R., 1976. The Complexity of Flowshop and Jobshop Scheduling. Math. Oper. Res. 1, 117–129.

Grossmann, I., 2005. Enterprise-wide optimization: A new frontier in process systems engineering. AIChE J. 51, 1846–1857.

Grossmann, I.E., 2012. Advances in mathematical programming models for enterprise-wide optimization. Comput. Chem. Eng. 47, 2–18.

Hadera, H., Harjunkoski, I., Sand, G., Grossmann, I.E., Engell, S., 2015. Optimization of steel production scheduling with complex time-sensitive electricity cost. Comput. Chem. Eng. 76, 117–136.

Hadera, H., Labrik, R., Sand, G., Engell, S., Harjunkoski, I., 2016. An Improved Energy-Awareness Formulation for General Precedence Continuous-Time Scheduling Models. Ind. Eng. Chem. Res. 55, 1336–1346.

Haït, A., Artigues, C., 2011. On electrical load tracking scheduling for a steel plant. Comput. Chem. Eng. 35, 3044–3047.

Harjunkoski, I., Bauer, R., 2017. Industrial scheduling solution based on flexible heuristics. Comput. Chem. Eng. 106, 883–891.

Harjunkoski, I., Grossmann, I.E., 2002. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. Comput. Chem. Eng. 26, 1533–1552.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J., Méndez, C., Sand, G., Wassick, J., 2014a. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193.

Harjunkoski, I., Maravelias, C.T., Bongers, P., Castro, P.M., Engell, S., Grossmann, I.E., Hooker, J.,

Méndez, C., Sand, G., Wassick, J., 2014b. Scope for industrial applications of production scheduling models and solution methods. Comput. Chem. Eng. 62, 161–193.

Hurink, J., Jurisch, B., Thole, M., 1994. Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spektrum 15, 205–215.

Ierapetritou, M.G., Floudas, C.A., 1998. Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. Ind. Eng. Chem. Res. 37, 4341–4359.

Jain, V., Grossmann, I.E., 2001. Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. INFORMS J. Comput. 13, 258–276.

Karwan, M.H., Keblis, M.F., 2007. Operations planning with real time pricing of a primary input. Comput. Oper. Res. 34, 848–867.

Klemmt, A., Horn, S., Weigert, G., Wolter, K.J., 2009. Simulation-based optimization vs. mathematical programming: A hybrid approach for optimizing scheduling problems. Robot. Comput. Integr. Manuf. 25, 917–925.

Koh, S., Eom, C., Jang, J., Choi, Y., 2008. An Improved Spatial Scheduling Algorithm for Block Assembly Shop in Shipbuilding Company, in: 2008 3rd International Conference on Innovative Computing Information and Control. IEEE, pp. 253–253.

Kopanos, G.M., Laínez, J.M., Puigjaner, L., 2009. An Efficient Mixed-Integer Linear Programming Scheduling Framework for Addressing Sequence-Dependent Setup Issues in Batch Plants. Ind. Eng. Chem. Res. 48, 6346–6357.

Kopanos, G.M., Méndez, C.A., Puigjaner, L., 2010. MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. Eur. J. Oper. Res. 207, 644–655.

Kulkarni, K., Venkateswaran, J., 2015. Hybrid approach using simulation-based optimisation for job shop scheduling problems. J. Simul. 9, 312–324.

Law, A.M., 2007. Simulation Modeling & Analysis, McGraw-Hili.

Lee, H., Maravelias, C.T., 2017. Mixed-integer programming models for simultaneous batching and

scheduling in multipurpose batch plants. Comput. Chem. Eng. 106, 621–644.

Lee, K., Shin, J.G., Ryu, C., 2009. Development of simulation-based production execution system in a shipyard: A case study for a panel block assembly shop. Prod. Plan. Control 20, 750–768.

Lee, T.S., Loong, Y.T., 2019. A review of scheduling problem and resolution methods in flexible flow shop. Int. J. Ind. Eng. Comput. 10, 67–88.

Li, Z., Ierapetritou, M., 2008. Process scheduling under uncertainty: Review and challenges. Comput. Chem. Eng. 32, 715–727.

Lim, M.-F., Karimi, I.A., 2003. Resource-Constrained Scheduling of Parallel Production Lines Using Asynchronous Slots. Ind. Eng. Chem. Res. 42, 6832–6842.

Lin, X., Floudas, C.A., 2001. Design, synthesis and scheduling of multipurpose batch plants via an effective continuous-time formulation. Comput. Chem. Eng. 25, 665–674.

Liu, Z., Chua, D.K.H., Wee, K.H., 2011. A Simulation Model for Spatial Scheduling of Dynamic Block Assembly in Shipbuilding. J. Eng. Proj. Prod. Manag. 1, 3–12.

Longo, F., 2013. On the short period production planning in industrial plants: a real case study. Int. J. Simul. Process Model. 8, 17.

Majozi, T., Seid, E.R., Lee, J.-Y., 2017. Understanding Batch Chemical Processes, 1st Editio. ed. CRC Press, Boca Raton.

Maravelias, C.T., 2012. General Framework and Modeling Approach Classification for Chemical Production Scheduling. AIChe J. 58, 1812–1828.

Maravelias, C.T., Grossmann, I.E., 2004. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. Comput. Chem. Eng. 28, 1921–1949.

Maravelias, C.T., Sung, C., 2009. Integration of production planning and scheduling: Overview, challenges and opportunities. Comput. Chem. Eng. 33, 1919–1930.

Méndez, C.., Henning, G.., Cerdá, J., 2000. Optimal scheduling of batch plants satisfying multiple product orders with different due-dates. Comput. Chem. Eng. 24, 2223–2245.

Méndez, C.A., Cerdá, J., 2003. An MILP Continuous-Time Framework for Short-Term Scheduling of Multipurpose Batch Processes Under Different Operation Strategies. Optim. Eng. 4, 7–22.

Méndez, C.A., Cerdá, J., 2003. Dynamic scheduling in multiproduct batch plants. Comput. Chem. Eng. 27, 1247–1259.

Méndez, C.A., Cerdá, J., 2002. An MILP framework for Short-Term Scheduling of Single-Stage Batch Plants with Limited Discrete Resources, in: Physics Letters A. pp. 721–726.

Méndez, C.A., Cerdá, J., Grossmann, I.E., Harjunkoski, I., Fahl, M., 2006. State-of-the-art review of optimization methods for short-term scheduling of batch processes. Comput. Chem. Eng. 30, 913–946.

Méndez, C.A., Henning, G.P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Comput. Chem. Eng. 25, 701–711.

Méndez, C.A., Henning, G.P., Cerdá, J., 2001. An MILP continuous-time approach to short-term scheduling of resource-constrained multistage flowshop batch facilities. Comput. Chem. Eng. 25, 701–711.

Mitra, S., Grossmann, I.E., Pinto, J.M., Arora, N., 2012. Optimal production planning under time-sensitive electricity prices for continuous power-intensive processes. Comput. Chem. Eng. 38, 171–184.

Mitra, S., Pinto, J.M., Grossmann, I.E., 2014a. Optimal multi-scale capacity planning for power-intensive continuous processes under time-sensitive electricity prices and demand uncertainty. Part II: Enhanced hybrid bi-level decomposition. Comput. Chem. Eng. 65, 102–111.

Mitra, S., Pinto, J.M., Grossmann, I.E., 2014b. Optimal multi-scale capacity planning for power-intensive continuous processes under time-sensitive electricity prices and demand uncertainty. Part I: Modeling. Comput. Chem. Eng. 65, 89–101.

Mitra, S., Sun, L., Grossmann, I.E., 2013. Optimal scheduling of industrial combined heat and power plants under time-sensitive electricity prices. Energy 54, 194–211.

Nolde, K., Morari, M., 2010. Electrical load tracking scheduling of a steel plant. Comput. Chem. Eng. 34, 1899–1903.

Osman, I.H., Kelly, J.P., 1996. Meta-Heuristics Theory and Applications, Springer Science & Business Media. Springer Science & Business Medi.

Pacciarelli, D., 2002. Alternative graph formulation for solving complex factory-scheduling problems. Int. J. Prod. Res. 40, 3641–3653.

Pan, C.-H., 1997. A study of integer programming formulations for scheduling problems. Int. J. Syst. Sci. 28, 33–41.

Pan, Q.-K., Ruiz, R., 2013. A comprehensive review and evaluation of permutation flowshop heuristics to minimize flowtime. Comput. Oper. Res. 40, 117–128.

Pattison, R.C., Touretzky, C.R., Johansson, T., Harjunkoski, I., Baldea, M., 2016. Optimal Process Operations in Fast-Changing Electricity Markets: Framework for Scheduling with Low-Order Dynamic Models and an Air Separation Application. Ind. Eng. Chem. Res. 55, 4562–4584.

Pinedo, M.L., 2016. Scheduling: Theory, Algorithms, and Systems, Fifth Edit. ed, Scheduling: Theory, Algorithms, and Systems. Springer, New York.

Pinedo, M.L., 2008. Scheduling: Theory, algorithms, and systems, Third Edit. ed, Scheduling: Theory, Algorithms, and Systems. Springer, NewYork, NY.

Pinto, J.M., Grossmann, I.E., 1998. Assignment and sequencing models for thescheduling of process systems. Ann. Oper. Res. 81, 433–466.

Pinto, J.M., Grossmann, I.E., 1995a. A Continuous Time Mixed Integer Linear Programming Model for Short Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res 34, 3037–3051.

Pinto, J.M., Grossmann, I.E., 1995b. A Continuous Time Mixed Integer Linear Programming Model for Short Term Scheduling of Multistage Batch Plants. Ind. Eng. Chem. Res. 34, 3037–3051.

Rippin, D.W.T., 1996. Batch Processing Systems Engineering. Springer Berlin Heidelberg, Berlin, Heidelberg.

Roshanaei, V., Azab, A., Elmaraghy, H., 2013. Mathematical modelling and a meta-heuristic for flexible job shop scheduling. Int. J. Prod. Res. 51, 6247–6274.

Roslöf, J., Harjunkoski, I., Björkqvist, J., Karlsson, S., Westerlund, T., 2001. An MILP-based reordering algorithm for complex industrial scheduling and rescheduling. Comput. Chem. Eng. 25, 821–828.

Roslöf, J., Harjunkoski, I., Westerlund, T., Isaksson, J., 2002. Solving a large-scale industrial scheduling problem using MILP combined with a heuristic procedure. Eur. J. Oper. Res. 138, 29–42.

Roslöf, J., Harjunkoski, I., Westerlund, T., Isaksson, J., 1999. A short-term scheduling problem in the paper-converting industry. Comput. Chem. Eng. 23, S871–S874.

Rossi, A., 2014. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. Int. J. Prod. Econ. 153, 253–267.

Ruiz, R., Maroto, C., 2005. A comprehensive review and evaluation of permutation flowshop heuristics. Eur. J. Oper. Res. 165, 479–494.

Ruiz, R., Stützle, T., 2008. An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. Eur. J. Oper. Res. 187, 1143–1159.

Saidi-Mehrabad, M., Fattahi, P., 2007. Flexible job shop scheduling with tabu search algorithms. Int. J. Adv. Manuf. Technol. 32, 563–570.

Shang, Z., Gu, J., Ding, W., Duodu, E.A., 2017. Spatial Scheduling Optimization Algorithm for Block Assembly in Shipbuilding. Math. Probl. Eng. 2017, 1–10.

Shen, L., Dauzère-Pérès, S., Neufeld, J.S., 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. Eur. J. Oper. Res. 265, 503–516.

Tamssaouet, K., Dauzère-Pérès, S., Yugma, C., 2018. Metaheuristics for the job-shop scheduling problem with machine availability constraints. Comput. Ind. Eng. 125, 1–8.

Voudouris, V.T., Grossmann, I.E., 1993. Optimal Synthesis of Multiproduct Batch Plants with Cyclic

Scheduling and Inventory Considerations. Ind. Eng. Chem. Res. 32, 1962–1980.

Wainer, G., 2009. Discrete-Event Modeling and Simulation, Computational Analysis, Synthesis, & Design Dynamic Systems. CRC Press.

Wolfsmayr, U.J., Merenda, R., Rauch, P., Longo, F., Gronalt, M., 2015. Evaluating primary forest fuel rail terminals with discrete event simulation: A case study from Austria. Ann. For. Res. 59, 1.

Xia, W., Wu, Z., 2005. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput. Ind. Eng. 48, 409–425.

Xiong, F., Xing, K., Wang, F., 2015. Scheduling a hybrid assembly-differentiation flowshop to minimize total flow time. Eur. J. Oper. Res. 240, 338–354.

Zamarripa, M., Marchetti, P.A., Grossmann, I.E., Singh, T., Lotero, I., Gopalakrishnan, A., Besancon, B., André, J., 2016. Rolling Horizon Approach for Production-Distribution Coordination of Industrial Gases Supply Chains. Ind. Eng. Chem. Res. 55, 2646–2660.

Zandieh, M., Gholami, M., 2009. An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. Int. J. Prod. Res. 47, 6999–7027.

Zeballos, L.J., Castro, P.M., Mèndez, C.A., 2011. Integrated constraint programming scheduling approach for automated wet-etch stations in semiconductor manufacturing. Ind. Eng. Chem. Res. 50, 1705–1715.

Zhang, L., Jiang, Y., Gao, X., Huang, D., Wang, L., 2016. Efficient Two-Level Hybrid Algorithm for the Refinery Production Scheduling Problem Involving Operational Transitions. Ind. Eng. Chem. Res. 55, 7768–7781.

Zhang, Q., Cremer, J.L., Grossmann, I.E., Sundaramoorthy, A., Pinto, J.M., 2016a. Risk-based integrated production scheduling and electricity procurement for continuous power-intensive processes. Comput. Chem. Eng. 86, 90–105.

Zhang, Q., Grossmann, I.E., 2016. Enterprise-wide optimization for industrial demand side management: Fundamentals, advances, and perspectives. Chem. Eng. Res. Des. 116, 114–131.

Zhang, Q., Sundaramoorthy, A., Grossmann, I.E., Pinto, J.M., 2016b. A discrete-time scheduling model for continuous power-intensive process networks with various power contracts. Comput. Chem. Eng. 84, 382–393.