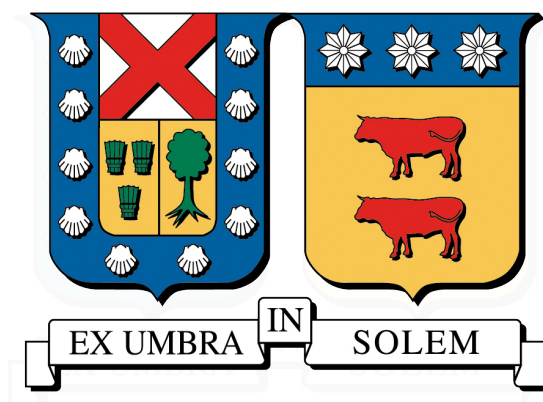


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INDUSTRIAS
SANTIAGO - CHILE



**PROGRAMACIÓN DE LA PRODUCCIÓN A TRAVÉS DE UN
MODELO DE PROGRAMACIÓN ENTERA-MIXTA PARA UNA
INDUSTRIA DE BEBIDAS NO ALCOHÓLICAS**

JAVIER ANDRÉS CUZMAR LEIVA

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA INDUSTRIAL
Y AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

PROFESOR GUÍA	:	SR. VÍCTOR M. ALBORNOZ
PROFESOR CORREFERENTE	:	SR. RODRIGO A. CARRASCO
PROFESORA CORREFERENTE	:	SRTA. MÓNICA A. LÓPEZ

AGOSTO 2016

Agradecimientos

Quiero expresar mis más sinceros agradecimientos a todas las personas que me acompañaron en este largo proceso, el cual culmina con el presente trabajo. Agradecer el apoyo, los ánimos, la paciencia, las ideas, los aportes y también los momentos de relajo.

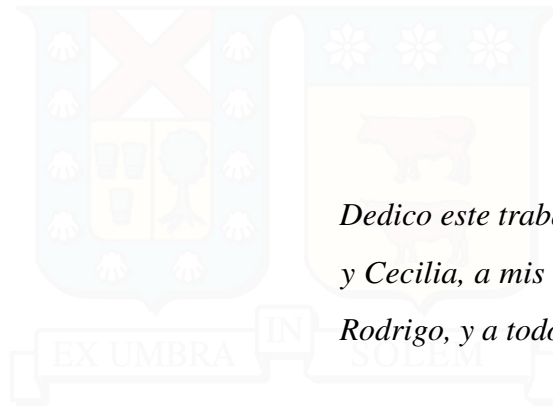
Primero agradecer a mi familia, a mis padres Sergio y Cecilia, pilares fundamentales en mi formación como persona, quienes siempre me apoyaron incondicionalmente. También agradecer a mis hermanos Macarena y Rodrigo, por sus consejos y tiempo.

Agradezco también a mis amigos, quienes compartieron conmigo durante mis años de universidad y vivimos momentos inolvidables, a ustedes amigos, simplemente gracias totales. Quisiera hacer una mención especial a mis amigos del magíster, con quienes compartí los últimos años de universidad, a Cristóbal, Marcelo y José.

Quisiera también agradecer a los trabajadores de ECCUSA, quienes me ayudaron en la elaboración de este trabajo y me dedicaron parte de su tiempo. Entre ellos me gustaría mencionar a Adrian, Natalia, Alejandra, José Miguel, Javier, Cristian y Constanza.

También quiero agradecer a mi profesor guía Víctor Albornoz, quien dedicó gran parte de su tiempo a dar forma a este trabajo, así como también a la Dirección General de Investigación y Postgrado y al departamento de Industrias de la Universidad Técnica Federico Santa María, que contribuyeron con financiamiento parcial a la investigación desarrollada.

A todos ustedes, muchas gracias.



Dedico este trabajo a mis padres Sergio y Cecilia, a mis hermanos, Macarena y Rodrigo, y a todos mis amigos.

RESUMEN EJECUTIVO

En este trabajo se aborda la formulación de un modelo de programación lineal entera-mixta para la programación de la producción en la industria de bebidas no alcohólicas. El modelo considera la producción de bebidas de diferentes sabores y formatos en dos etapas sincronizadas de producción: preparación del jarabe en tanques de almacenamiento y embotellado del líquido en líneas de envasado. Dicho modelo define el orden de los productos a fabricar en cada etapa de producción, minimizando el tiempo total de proceso de las ordenes a programar (Makespan minimization), tomando en cuenta aspectos como tiempos de set up dependientes de la secuencia, sincronización de las etapas de producción, varios tanques y líneas de envasado, restricciones de capacidad, restricciones de tiempo (deadlines) y la propiedad de división de lotes en la primera etapa, lo que permite maximizar la utilización de las líneas de envasado. Debido a la complejidad del modelo propuesto, uno de naturaleza NP-hard, la tesis contempla igualmente una estrategia de resolución en dos etapas, la primera de las cuales emplea el modelo para un problema de menor tamaño y, en la segunda, hace uso de una heurística o regla de decisión para fijar algunas decisiones. El trabajo incluye la aplicación a instancias de un problema real de una empresa embotelladora de bebidas. Los resultados muestran que, en promedio, se logró reducir en un 15,67 % el tiempo total de producción actual de la compañía.

Palabras Clave. Programación, Minimización del Tiempo de Producción, Programación Entera Mixta, Bebidas no Alcohólicas.

ABSTRACT

In this study, the formulation of a mixed-integer linear programming model applied to production scheduling in the soft drink industry is addressed. The model considers the production of beverages with different flavors and formats in two synchronized production stages: preparation of syrup in storage tanks and bottling syrup in packaging lines. This model defines the order of the products at each stage of production with makespan minimization, taking into account aspects such as sequence-dependent set-up times, synchronisation between production stages, several tanks and packaging lines, capacity constraints and time constraints (deadlines). Also considered is the property of job splitting in first stage, which reduces waiting times in the packaging lines. Due to the complexity of the model, generally classified as NP-Hard, it is presented a strategy for solution in two stages, the first of which uses the model to a smaller problem and, in the second, uses a heuristic or decision rule to set some decisions. This study include the application on a real-world problem of a beverage bottling company. The results show that on average the application managed to improve 15.67 % the company's current solution.

Keywords. Scheduling, Makespan, Mixed Integer Programing, Soft Drink.

Índice de Contenidos

1. Introducción	1
1.1. Contexto	2
1.2. Objetivos	5
1.2.1. General	5
1.2.2. Específicos	5
1.3. Motivación	6
1.3.1. Importancia y aporte sobre las bases	6
1.4. Estructura	7
2. Marco Teórico	8
2.1. Antecedentes de la programación de la producción (Scheduling)	10
2.1.1. Clasificación de la programación de tres campos	11
2.1.1.1. Primer campo α	11
2.1.1.2. Segundo campo β	13
2.1.1.3. Tercer campo γ	17
2.1.2. Programación de la producción basado en el Problema del Vendedor Viajero (TSP)	18
2.1.3. Ejemplos de la literatura de modelos basados en el TSP	21
2.2. Estado del Arte	26
3. Metodología	30
3.1. Descripción del problema: El proceso productivo	30
3.2. Modelo propuesto	34
3.3. Estrategia de aplicación del modelo	41
3.3.1. División del problema en clusters	41
3.3.2. Programación de productos a priori	41
4. Resultados	44
4.1. Aplicación al caso de estudio	44
4.2. Análisis de resultados	48
5. Conclusiones	53
5.1. Investigaciones futuras	54
Bibliografía	56

A. Código AMPL (.mod)**59**

Índice de Tablas

1.1. Producción y ventas de CCU a través de los años. (Fuente: Memoria anual CCU 2015.)	5
4.1. División de clusters para el caso de estudio.	45
4.2. Tamaño de las instancias de los clusters en cada semana.	46
4.3. Resultados computacionales para las instancias del caso de estudio. . . .	47

Índice de Figuras

2.1. Diagrama del flujo de información en un sistema de manufactura.	9
2.2. Posibles soluciones del TSP.	20
3.1. Proceso productivo de la industria de bebidas no alcohólicas.	31
3.2. Sincronización entre las etapas productivas.	33
4.1. Soluciones del criterio por tiempo de entrega.	49
4.2. Soluciones del criterio por tiempo de proceso.	49
4.3. Cantidad de productos programados a priori en cada instancia.	50
4.4. Sensibilidad del modelo en el tiempo de resolución con respecto a la cantidad de productos fijados a priori.	51
4.5. Sensibilidad del modelo en el tiempo de resolución con respecto al GAP utilizado.	51

1 | Introducción

La industria de bebidas no alcohólicas está presente en cada mercado de consumo con una gran variedad de productos como bebidas carbonatadas, aguas, jugos y bebidas funcionales, cada una en diferentes sabores y formatos. Las plantas de producción deben enviar los productos terminados a través de un vasto territorio a una gran cantidad de puntos de ventas, por lo que sus niveles de producción son bastante altos. Esta situación obliga a las plantas a producir de manera más efectiva sus productos. Un problema recurrente para ellas corresponde a convertir el plan agregado de producción (mediano plazo) a las ordenes de producción que determinan una programación detallada (corto plazo) en la que se especifica el orden en que son elaborados los productos en cada una de las etapas del mismo, por lo que poseer herramientas para su gestión puede ser una gran ventaja frente a los competidores.

En la industria de bebidas no alcohólicas, un aspecto que toma gran relevancia es la existencia de tiempos de set up dependientes de la secuencia, es decir, los tiempos de set up dependen del último producto elaborado, por lo que el orden en que se producen los productos tiene gran impacto en el tiempo total de producción. A grandes rasgos, el proceso productivo de esta industria consta de dos etapas principales: elaboración del jarabe y envasado. En la elaboración del jarabe, se utilizan tanques de almacenamiento en los cuales se prepara el líquido del producto. En la etapa de envasado, el líquido es transportado desde los tanques de almacenamiento hasta las líneas de envasado donde se llenan los envases con el líquido correspondiente.

En este trabajo se presenta la formulación de un modelo de programación entera mixta

que representa el proceso productivo mencionado, el cual determina el orden en que se producen los productos con el objetivo de minimizar el tiempo total de producción en un período de corto plazo. Las principales características del modelo son la incorporación de tiempos de set up dependientes de la secuencia, sincronización de las etapas de producción, varios tanques y líneas de envasado, restricciones de capacidad, restricciones de tiempo (deadlines) y la opción de dividir los lotes de producción entre varios tanques, lo que permite maximizar la utilización de las líneas de envasado cuando un lote supera la capacidad del tanque.

1.1. Contexto

El problema a resolver consiste en un proceso fundamental en la operación de una empresa, que está ligada a la planificación de la producción. Corresponde a la última y más detallada etapa: La programación de la producción. En esta etapa se establece la coordinación, seguimiento y control de las actividades (semanales, diarias, turnos, etc.) utilizando los procedimientos de asignación, secuenciación y temporización de la producción adecuados al tipo de proceso productivo que se desarrolle en cada empresa.

En particular, el presente trabajo se desarrolla en una empresa embotelladora de bebidas no alcohólicas, cuyo proceso de programación de la producción tiene ciertas características específicas. Una de las más importantes consideraciones a tomar en cuenta al realizar la programación corresponde a la existencia de dos etapas sincronizadas de producción. La primera etapa corresponde al almacenamiento de jarabe en tanques de almacenamiento y la segunda al envasado del jarabe en botellas en las líneas de envasado. Debido a esta característica, se deben programar los tanques y la línea de envasado que serán utilizados para cada producto, es decir, la asignación de máquinas. Para el secuenciamiento de productos, es importante tomar en cuenta que los tiempos de set up son dependientes de la secuencia en ambas etapas, debido a las características del proceso productivo de esta industria. También se deben computar los tiempos en que son completados los productos,

ya que existen tiempos de entrega que hay que cumplir.

Todas estas características deben ser tomadas en cuenta para la programación de la producción, de manera de lograr la mayor utilización de las máquinas, lo que se obtiene en este trabajo minimizando el tiempo total de proceso de los productos a elaborar.

La empresa en la cual se desarrolla este trabajo corresponde a Compañía Cervecerías Unidas S.A. (CCU), específicamente en su unidad de negocio llamada Embotelladoras Chilenas Unidas S.A. (ECUSA), donde se embotellan las bebidas no alcohólicas a excepción de aguas minerales. Actualmente, CCU es una sociedad anónima abierta, cuyas acciones son transadas en los mercados bursátiles de Chile y en la Bolsa de Valores de Nueva York. Sus principales accionistas son Quiñenco S.A., holding de la familia Luksic, y Heineken, quienes, a través de Inversiones y Rentas S. A., poseen el 60 % de la propiedad de la Compañía. El resto de las acciones están divididas entre ADR's (13.8 %) y otros accionistas minoritarios (26.2 %).

Productos

CCU cuenta con Unidades de Negocio y Servicio bien definidas. Dentro de las Unidades de negocio se encuentran Cervecera CCU Chile Ltda., Embotelladoras Chilenas Unidas S.A., Aguas CCU Nestlé S.A., Compañía Pisquera de Chile S.A., Viña San Pedro Tarapacá S.A. y CCU Argentina S.A. Dentro de las Unidades de Servicio se encuentran la distribución y producción de envases como Comercial CCU S.A., Transportes CCU Ltda. y la Fábrica de Envases Plásticos PLASCO S.A. Dentro de Embotelladoras Chilenas Unidas S.A. existe gran variedad de marcas como Bilz y Pap, Kem, Pepsi, Crush, Canada Dry, Watts, Adrenaline, Gatorade, entre otros.

Clientes

En Chile, la fuerza de ventas de CCU está integrada por más de mil personas que

atienden en forma directa a más de 160 mil clientes. La comercialización de sus productos está dirigida básicamente al comercio minorista, integrado por supermercados, restaurantes, botillerías, fuentes de soda y otros. De esta forma, CCU atiende a través de sus filiales alrededor de 115.179 clientes en Chile. Sus principales clientes son las cadenas de supermercados Walmart y Cencosud, con mayor poder de negociación. Otros clientes en Chile de CCU son Hipermercado Tottus, Monserrat, Alvi Supermercados Mayoristas y Ekono, entre otros, con menor participación.

Competidores y participación de mercado

Para hablar de participación, es necesario separar los distintos mercados debido a la diversidad de los productos que ofrece CCU, por lo que se detallará su participación con respecto al mercado de bebidas no alcohólicas. En el caso de gaseosas (excluyendo funcionales y té), CCU es el segundo mayor embotellador con aproximadamente 24,8 % del mercado. Los embotelladores de Coca-Cola (72 % Embotelladora Andina S.A. y 28 % Embonor S.A.) poseen una participación de alrededor de 68,2 %. En el segmento de los néctares en botellas, la Compañía, con la marca Watt's, es líder con 55,5 % de participación de mercado, siendo su principal competidor Andina con 31,3 % de participación

Estadísticas de producción y ventas

En 2015, los Ingresos por ventas de Compañía Cervecerías Unidas S.A. (CCU) aumentaron 15,4 %, alcanzando a CLP 1.498.372 millones, como consecuencia de volúmenes consolidados 4,4 % más altos y precios promedio 10,6 % más altos. El Estado Consolidado de Resultados por Función presenta un crecimiento del Margen Bruto de 17,3 %, alcanzando a CLP 813.296 millones, como resultado del aumento de 15,4 % de los Ingresos por ventas, parcialmente contrarrestado por un Costo de ventas 13,3 % mayor. El Costo de ventas como porcentaje de los Ingresos por ventas, disminuyó de 46,6 % a 45,7 %. Como consecuencia, el Margen Bruto como porcentaje de los Ingresos por ventas aumentó de 53,4 % a 54,3 %.

El Resultado operacional (o EBIT por sus siglas en inglés) aumentó 13,9 % alcanzando CLP 204.937 millones, y el EBITDA aumentó 15,3 % alcanzando CLP 286.504 millones. Por otra parte, el Resultado no operacional disminuyó CLP 1.545 millones, de una pérdida de CLP 12.752 millones a una pérdida de CLP 14.297 millones, debido principalmente a mayores Gastos financieros netos y menores Utilidad de negocios conjuntos y asociadas.

Tabla 1.1: Producción y ventas de CCU a través de los años.

(Fuente: Memoria anual CCU 2015.)

EERR Resumido (MMS)	2011	2012	2013	2014	2015
Ingresos por Ventas	969.551	1.075.690	1.197.277	1.297.966	1.498.372
EBIT	192.818	181.188	188.266	179.920	204.937
Mg EBIT (%)	19,9	16,8	15,7	13,9	13,7
Utilidad Neta	122.752	114.433	123.036	119.557	120.808
EBITDA	240.600	235.948	252.512	248.528	286.504
Mg EBITDA (%)	24,8	21,9	21,1	19,1	19,1
Volumen (Miles de Hectolitros)	18.397	19.909	21.916	22.898	23.910

1.2. Objetivos

1.2.1. General

El objetivo general de este trabajo es **formular un modelo de programación lineal entera-mixta y una estrategia de resolución basado en el mismo, que determine la programación detallada de la producción para el proceso de elaboración de bebidas no-alcohólicas en su aplicación a un caso de naturaleza real de una empresa embotelladora**. Esto tiene como fin establecer una metodología de trabajo periódica que permita utilizar de mejor manera la planta productiva y así optimizar el proceso de programación de la producción.

1.2.2. Específicos

Los objetivos específicos de este trabajo son:

- Revisar literatura con respecto al problema.
- Identificar los procesos y recopilar datos y antecedentes a modelar.
- Desarrollar un modelo de optimización que represente de mejor manera la problemática a abordar.
- Determinar una estrategia de resolución del modelo propuesto para el caso de estudio a presentar.
- Evaluar el desempeño de la metodología utilizada para el problema real abordado.

1.3. Motivación

1.3.1. Importancia y aporte sobre las bases

El problema desarrollado en el presente trabajo consiste en optimizar el proceso de programación de la producción, es decir, utilizar herramientas de la investigación de operaciones para mejorar un proceso en las compañías, específicamente compañías en la industria de bebidas no alcohólicas.

La literatura muestra algunos artículos referidos también a la industria de bebidas no alcohólicas, pero de acuerdo a los trabajos consultados, ninguno enfocado exclusivamente en la programación de la producción a corto plazo. Este trabajo puede ser aplicado e instaurado como una re-definición del proceso de programación de la producción. El objetivo del modelo propuesto corresponde a la minimización del tiempo total de proceso de los productos a programar. Si al utilizar esta metodología se logra reducir este tiempo, con respecto al actual de una compañía, esta puede planificar su producción con mayores cantidades para el período ejecutado, lo que implica mayor producción en el mismo período. Sin duda esto significa una mejora en la operación de una compañía, por lo que poseer herramientas para su gestión puede ser una gran ventaja frente a los competidores.

1.4. Estructura

En adelante el presente informe está organizado de la siguiente manera. En el Capítulo 2 se presenta el marco teórico, donde primero se entregan antecedentes teóricos de la programación de la producción, para luego explicar los detalles en los cuales se basó este trabajo y ciertas características utilizadas para el modelo propuesto. También se resume la revisión de la literatura enfocada en trabajos relacionados. En el Capítulo 3 se aborda la metodología a seguir, donde se describe la problemática a abordar, detallando el proceso productivo de la industria de bebidas no alcohólicas, para luego presentar el modelo matemático propuesto y su estrategia de resolución. En el Capítulo 4 se presentan los resultados computacionales aplicados a un problema real de una empresa embotelladora. Finalmente, en el Capítulo 5 se muestran las principales conclusiones, junto con una mirada a investigaciones futuras.

2 | Marco Teórico

El ámbito en el que se desenvuelve este trabajo corresponde a un proceso muy importante de la operación de una empresa manufacturera, llamada la programación de la producción. Las ordenes que son emitidas en un escenario de manufactura tienen que ser transformadas en trabajos (jobs) con un tiempo de entrega asociado. Estos trabajos generalmente tienen que ser procesados en máquinas en una estación de trabajo en un orden o secuencia dada. El procesamiento de los trabajos a veces puede retrasarse si ciertas máquinas están ocupadas y también pueden ocurrir preferencias de producción cuando llegan trabajos de alta prioridad a una máquina que está ocupada. Existen tiempos de preparación de las máquinas para el procesamiento de un trabajo (setup times) que deben realizarse cuando éste llega, el cual puede depender o no del trabajo a procesar. Las máquinas pueden ser diferentes en velocidad, capacidad y/o variedad de procesos. En tal entorno, el desarrollo de una detallada programación de tareas ayuda a mantener la eficiencia y control de las operaciones.

El control de planta o producción (Shop floor) no es la única parte de la organización que impacta en el proceso de programación. También es afectado por el proceso de planificación de la producción que maneja la planificación de mediano a largo plazo para toda la organización. Este proceso busca optimizar la producción total de la firma y la asignación de recursos a largo plazo basado en sus niveles de inventario, pronósticos de demanda y requerimiento de recursos. Las decisiones realizadas en este nivel de planificación pueden impactar directamente el proceso de programación. La Figura 2.1 describe un diagrama del flujo de información en un sistema de manufactura.

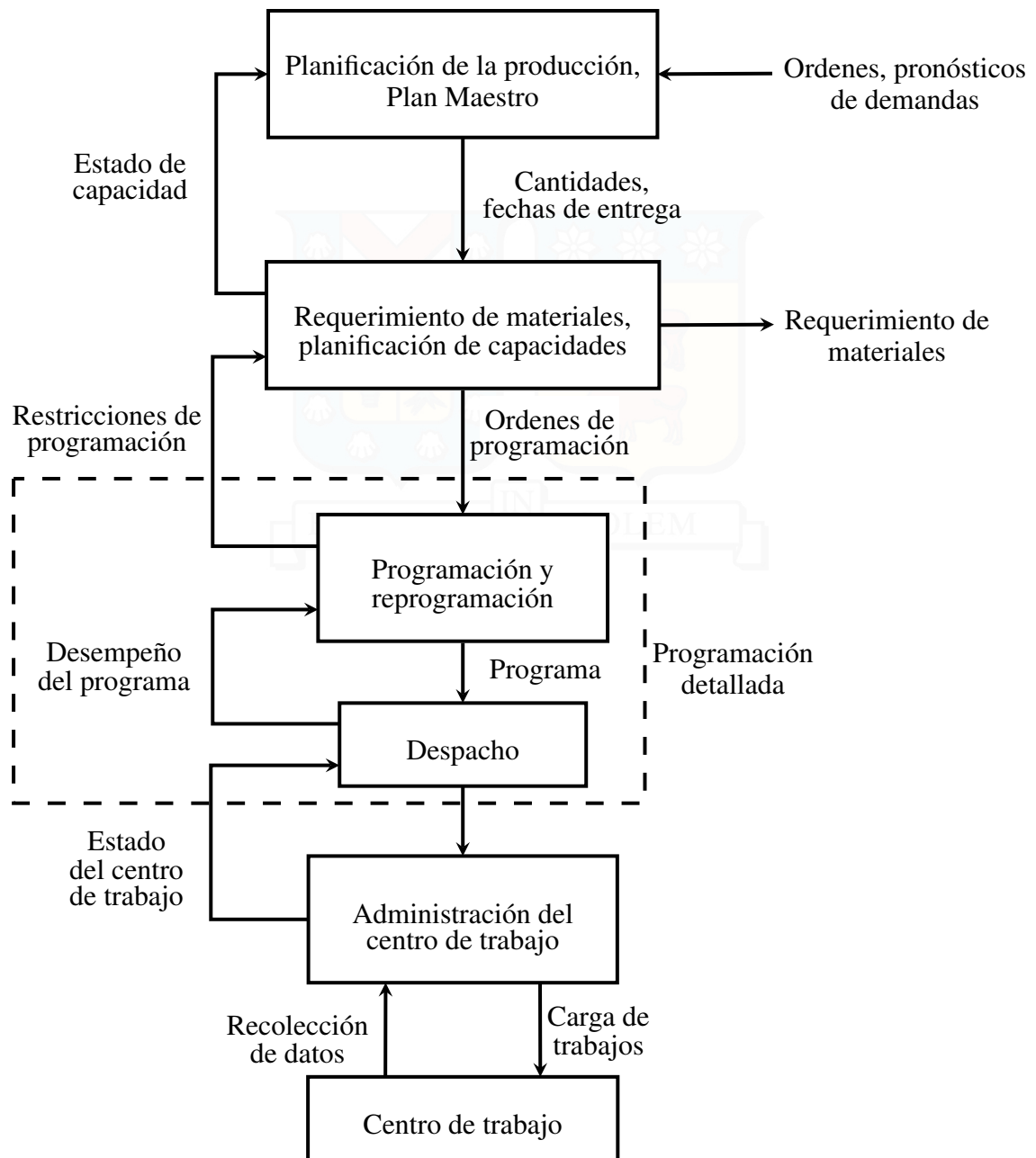


Figura 2.1: Diagrama del flujo de información en un sistema de manufactura.

(Fuente: Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media.)

En un entorno de manufactura, la función de la programación tiene que interactuar con otras funciones de decisión. Un sistema muy popular que es largamente utilizado es el sistema de Planificación de los Requerimientos de Material (MRP). Después de que se ha generado una programación es necesario que todas las materias primas y los recursos

estén disponibles en los tiempos especificados. Cada trabajo tiene una Lista de Materiales (BOM, por sus siglas en inglés Bill of Materials) que contiene las partes requeridas para su producción. El sistema MRP mantiene un seguimiento del inventario de cada parte. Más aún, determina el tiempo de adquisición de cada uno de los materiales. Al hacer esto, se utilizan técnicas como el dimensionamiento de lotes (Lot Sizing) y programación de lotes, que son similares a los usados en los sistemas de programación. Existen muchos paquetes de software comerciales de sistemas de MRP y, como resultado, hay muchas empresas manufactureras con sistemas MRP. En los casos en que la empresa no cuenta con sistemas de programación, el sistema MRP puede ser usado para propósitos de planificación de la producción. Sin embargo, para ajustes complejos no es fácil para un sistema MRP realizar una programación detallada satisfactoriamente.

2.1. Antecedentes de la programación de la producción (Scheduling)

A continuación se presentan algunos antecedentes pertenecientes al tópico de programación de la producción. Para más información ver [Pinedo \(2015\)](#).

En los problemas de programación considerados se asume que el número de trabajos y de máquinas es finito. El número de trabajos es denotado por n y el número de máquinas por m . Usualmente, el subíndice j se refiere a un trabajo mientras que el subíndice i se refiere a una máquina. Si un trabajo requiere un número de etapas de procesos u operaciones, entonces el par (i, j) se refiere al proceso del trabajo j en la máquina i . La siguiente información está asociada al trabajo j .

- **Processing time** (p_{ij}): Representa el tiempo de proceso del trabajo j en la máquina i . El subíndice i es omitido si el tiempo de proceso del trabajo j no depende de la máquina o si el trabajo j es sólo procesado en una máquina dada.
- **Release date** (r_j): Corresponde al tiempo en que el trabajo j llega al sistema, es decir, el tiempo más temprano en que el trabajo j puede empezar su procesamiento.

También es llamado Ready date.

- **Due date** (d_j): Representa el tiempo de entrega del trabajo j . La completación de un trabajo después de su tiempo de entrega está permitida, pero entonces se incurre en penalización. Cuando el tiempo de entrega *debe* ser cumplido se refiere a un *deadline*.
- **Weight** (w_j): Representa básicamente un factor de prioridad, denotando la importancia del trabajo j con respecto a los otros trabajos en el sistema. Por ejemplo, este factor puede representar el costo actual de mantener el trabajo en el sistema. Este costo puede ser un costo de inventario; también podría representar la cantidad de valor ya agregada al trabajo.

2.1.1. Clasificación de la programación de tres campos

Los problemas de programación se describirán por la combinación de tres campos diferentes: $\alpha/\beta/\gamma$, introducidos en [Graham et al. \(1979\)](#). El campo α describe el entorno de máquinas y contiene sólo una entrada. El campo β provee los detalles de las características del procesamiento y restricciones y puede no tener entrada, sólo una, o múltiples entradas. El campo γ describe el objetivo a ser minimizado y usualmente contiene sólo una entrada.

2.1.1.1. Primer campo α

Los posibles entornos de máquinas especificados en el campo α son:

- **Single machine** (1): El caso de una máquina es el más simple de todos los entornos de máquinas posibles y es un caso especial de otros entornos más complicados.
- **Identical machines in parallel** (Pm): Hay m máquinas idénticas en paralelo. El trabajo j requiere una única operación y puede ser procesado por cualquiera de las m máquinas o por cualquiera que pertenezca a un subconjunto dado.
- **Machines in parallel with different speeds** (Qm): Hay m máquinas en paralelo con diferentes velocidades. La velocidad de la máquina i es denotada por v_i . El tiempo p_{ij} que el trabajo j gasta en la máquina i es igual a p_j/v_i (asumiendo que el trabajo j es

procesado por completo por la máquina i). Este entorno es conocido como máquinas *uniformes*. Si todas las máquinas tienen la misma velocidad entonces el entorno es idéntico al anterior.

- **Unrelated machines in parallel (Rm):** Este entorno es una mayor generalización que el anterior. Hay m máquinas diferentes en paralelo. La máquina i puede procesar el trabajo j a una velocidad v_{ij} . El tiempo p_{ij} que un trabajo j gasta en la máquina i es igual a p_j/v_{ij} (de nuevo asumiendo que el trabajo j es procesado por completo por la máquina i). Si las velocidades de las máquinas son independientes de los trabajos, entonces el entorno es idéntico al anterior.
- **Flow shop (Fm):** Hay m máquinas en serie. Cada trabajo tiene que ser procesado en cada una de las m máquinas. Todos los trabajos tienen que seguir la misma ruta, es decir, tienen que ser procesados primero en la máquina 1, luego en la máquina 2, y así en adelante. Después de terminar en una máquina, un trabajo se une a la fila de la siguiente máquina. Usualmente, para todas las filas se asume que operan bajo la regla de First In First Out (FIFO), esto es, un trabajo no puede “pasar” a otro mientras espera en la fila. Si se utiliza la regla FIFO, este entorno es conocido como *permutation flow shop* y el campo β incluye la entrada *prmu*.
- **Flexible flow shop (FFc):** Este entorno es una generalización del entorno flow shop y parallel machine. En vez de m máquinas en serie hay c etapas en serie con un número de máquinas idénticas en paralelo en cada etapa. Cada trabajo tiene que ser procesado primero en la etapa 1, luego en la etapa 2, y así en adelante. Una etapa funciona como un banco de máquinas en paralelo; en cada etapa el trabajo j requiere ser procesado en sólo una máquina y cualquier máquina puede hacerlo. Las filas entre las etapas pueden o no pueden operar con la regla FIFO. Este entorno también es conocido en la literatura como *hybrid flow shops* y como *multi-processor flow shops*.
- **Job shop (Jm):** En este entorno con m máquinas, cada trabajo tiene su propia ruta específica a seguir. Existe una distinción entre job shops en el que cada trabajo visita

cada máquina al menos una vez y job shops en el que un trabajo puede visitar cada máquina más de una vez. En el último caso el campo β contiene la entrada *rcrc* de recirculación.

- **Flexible job shop (*FJc*)**: Este entorno es una generalización del entorno job shop y del parallel machine. En vez de m máquinas en serie hay c estaciones de trabajo con un número de máquinas idénticas en paralelo en cada estación de trabajo. Cada trabajo tiene su propia ruta específica a seguir; el trabajo j requiere procesamiento en cada estación de trabajo en sólo una máquina y cualquier máquina puede hacerlo. Si un trabajo en su ruta a seguir puede visitar una estación de trabajo más de una vez, entonces el campo β contiene la entrada *rcrc* de recirculación.
- **Open shop (*Om*)**: Hay m máquinas. Cada trabajo tiene que ser procesado de nuevo en cada una de las m máquinas. Sin embargo, algunos de estos tiempos de proceso pueden ser cero. No hay restricciones con respecto a la ruta de cada trabajo a través del entorno de máquinas. El programador tiene permitido determinar una ruta para cada trabajo y diferentes trabajos pueden tener diferentes rutas.

Para el caso del problema abordado, el ambiente de máquinas corresponde a un **flexible flow shop**, ya que se cuenta con dos etapas de producción con máquinas en paralelo en cada etapa. La única diferencia es que en este caso, las etapas operan sincronizadamente, lo que será explicado en detalle en la sección 3.1.

2.1.1.2. Segundo campo β

Las características de proceso y restricciones especificadas en el campo β pueden incluir múltiples entradas. Posibles entradas para el campo β son:

- **Release dates (r_j)**: Si este símbolo aparece en el campo β , entonces el trabajo j no puede empezar su procesamiento antes del tiempo r_j de llegada al sistema. Si r_j no aparece en el campo β , el procesamiento del trabajo j puede empezar en cualquier tiempo. En contraste con release date, due date no es especificado en este campo. El tipo de función objetivo entrega suficiente información de si hay o no due date.

- **Preemptions** (*prmp*): Este adelantamiento implica que no es necesario mantener un trabajo en una máquina. El programador tiene permitido interrumpir el procesamiento de un trabajo en cualquier punto del tiempo y poner un trabajo diferente en esa máquina. La cantidad de procesamiento de un trabajo reemplazado que ya fue realizado no se pierde. Cuando un trabajo que fue adelantado es después puesto de vuelta en la máquina (o en otra máquina en caso de máquinas en paralelo), sólo es necesario procesar el tiempo faltante para su término. Cuando preemptions está permitido, el campo β incluye la entrada *prmp*; cuando *prmp* no es incluido, preemptions no está permitido.
- **Precedence constraints** (*prec*): Restricciones de precedencia pueden aparecer en entornos de una máquina o en máquinas en paralelo, con la condición que uno o más trabajos deben ser completados antes que otro trabajo comience su procesamiento. Hay varias formas especiales de restricciones de precedencia: si cada trabajo tiene a lo más un predecesor y a lo más un sucesor, las restricciones son conocidas como *chains* (cadenas). Si cada trabajo tiene a lo más un sucesor, las restricciones son conocidas como *intree* (árbol de entrada). Si cada trabajo tiene a lo más un predecesor, las restricciones son conocidas como *outtree* (árbol de salida). Si no aparece *prec* en el campo β , los trabajos no están sujetos a restricciones de precedencia.
- **Sequence dependent setup times** (s_{jk}): Esto significa que los tiempos de setup son dependientes de la secuencia, es decir, el tiempo de setup de un trabajo depende del trabajo procesado anteriormente. s_{jk} representa el tiempo de setup dependiente de la secuencia en el que se incurre entre los procesamientos de los trabajos j y k ; s_{0k} denota el tiempo de setup para el trabajo k si es el primero en la secuencia y s_{j0} es el tiempo de limpieza después del trabajo j si es el último en la secuencia (por supuesto, s_{0k} y s_{j0} pueden ser cero). Si el tiempo de setup entre los trabajos j y k dependen de la máquina, entonces se debe incluir el subíndice i , es decir, s_{ijk} . Si s_{jk} no aparece en el campo β , todos los tiempos se asumen cero o independientes de la secuencia, en cuyo caso son simplemente incluidos en el tiempo de proceso.

- **Job families** ($fmls$): En este caso los n trabajos pertenecen a F diferentes familias de trabajos. Los trabajos de la misma familia pueden tener diferentes tiempos de proceso, pero pueden ser procesados en una máquina uno tras otro sin requerir tiempo de setup entre ellos. Sin embargo, si la máquina cambia de una familia a otra, por ejemplo, de la familia g a la familia h , entonces se requiere un tiempo de setup. Si este tiempo de setup depende de ambas familias y es dependiente de la secuencia, entonces se denota como s_{gh} . Si este tiempo de setup depende sólo de la familia que está por comenzar su proceso, por ejemplo la familia h , entonces es denotado por s_h . Si no depende de las familias, es denotado simplemente s .
- **Batch processing** ($batch(b)$): Una máquina puede ser capaz de procesar un número de trabajos b simultáneamente; esto es, puede procesar un lote de hasta b trabajos al mismo tiempo. El tiempo de proceso de los trabajos en un lote puede no ser para todos el mismo y el lote completo es terminado sólo cuando el último trabajo del lote ha sido terminado, lo que implica que el tiempo de completación del lote está determinado por el trabajo con mayor tiempo de proceso. Si $b = 1$, entonces el problema se reduce a un entorno convencional de programación. Otro caso especial de interés es cuando $b = \infty$, es decir, no hay límite para el número de trabajos que la máquina puede manejar.
- **Breakdowns** ($brkdwn$): Esto implica que una máquina puede no estar continuamente disponible. Se asume que los períodos que una máquina no está disponible pueden ser reparables (por ejemplo, debido a los turnos o mantenciones programadas). Si hay un número de máquinas idénticas en paralelo, el número de máquinas disponibles en cualquier punto del tiempo es una función del tiempo, por ejemplo, $m(t)$. Este caso también es conocido como machine availability constraints.
- **Machine eligibility restrictions** (M_j): Este símbolo puede aparecer en el campo β cuando el entorno de máquinas es m máquinas en paralelo (Pm). Cuando M_j está presente, no todas las m máquinas son capaces de procesar el trabajo j . El conjunto M_j define el conjunto de máquinas que pueden procesar el trabajo j . Si el campo β no contiene M_j , el trabajo j puede ser procesado en cualquiera de las m máquinas.

- **Permutation** (*prmu*): Una restricción que puede aparecer en el entorno flow shop es que las filas frente a cada máquina pueden operar según la regla First In First Out (FIFO). Esto implica que el orden (o permutación) en que los trabajos avanzan a través de la primera máquina se mantiene a través del sistema.
- **Blocking** (*block*): Este es un fenómeno que puede ocurrir en entorno flow shop. Si un flow shop tiene un buffer limitado entre dos máquinas sucesivas, entonces puede ocurrir que cuando el buffer está lleno, las máquinas aguas arriba no tienen permitido liberar un trabajo terminado. Este bloqueo implica que los trabajos completados tienen que permanecer en las máquinas aguas arriba bloqueándolas para procesar el siguiente trabajo. La ocurrencia más común de bloqueo es el caso en el que hay cero buffer entre dos máquinas cualquiera. En este caso un trabajo que a completado su procesamiento en una máquina dada no puede dejar la máquina si el trabajo que antecede aún no ha completado su procesamiento en la siguiente máquina; así, el trabajo bloqueado también bloquea el siguiente trabajo para comenzar su procesamiento en la máquina dada. Si las máquinas operan con FIFO, entonces *block* implica *prmu*.
- **No-wait** (*nwt*): Este es otro fenómeno que puede ocurrir en flow shops. Los trabajos no tienen permitido esperar entre dos máquinas sucesivas. Esto implica que el tiempo en que comienza un trabajo en la primera máquina tiene que ser retrasado para asegurar que el trabajo avance a través del flow shop sin tener que esperar por ninguna máquina. Un ejemplo de tal operación es un laminador de acero en el cual una losa de acero no tiene permitido esperar ya que podría enfriarse mientras espera. Es claro que cuando hay no-wait las máquinas también operan bajo la regla FIFO.
- **Recirculation** (*rcrc*): La recirculación puede ocurrir en job shop o flexible job shop cuando un trabajo puede visitar una máquina o estación de trabajo más de una vez.

La característica del proceso utilizada en este caso corresponde a la existencia de **sequence dependent setup times**, es decir, tiempos de setup dependientes de la secuencia, debido a las características del proceso productivo propias de la industria de bebidas no alcohólicas.

2.1.1.3. Tercer campo γ

El objetivo a ser minimizado es siempre una función de los tiempos de completación de los trabajos, los cuales, por supuesto, dependen de la programación. El tiempo de completación de operación del trabajo j en la máquina i se denota como C_{ij} . El tiempo en que el trabajo j permanece en el sistema (esto es, su tiempo de completación en la última máquina requerida para su procesamiento) se denota como C_j . El objetivo también puede ser función del tiempo de entrega (due date). Se define *lateness* (retraso) del trabajo j como

$$L_j = C_j - d_j,$$

lo que es positivo cuando el trabajo j es terminado tarde y negativo cuando es terminado temprano. Se define *tardiness* (tardanza) del trabajo j como

$$T_j = \max(C_j - d_j, 0) = \max(L_j, 0).$$

La diferencia entre *tardiness* y *lateness* es el hecho de que *tardiness* nunca es negativo. Se define *unit penalty* (penalización unitaria) del trabajo j como

$$U_j = \begin{cases} 1 & \text{si } C_j > d_j, \\ 0 & \text{en otro caso.} \end{cases}$$

Ejemplos de posibles funciones objetivos a ser minimizados denotados en el campo γ son:

- **Makespan** ($C_{\max}$): El makespan, definido como $\max(C_1, \dots, C_n)$, es equivalente al tiempo de completación del último trabajo en dejar el sistema. El mínimo makespan usualmente implica una buena utilización de la(s) máquina(s).
- **Maximum Lateness** ($L_{\max}$): Esto es definido como $\max(L_1, \dots, L_n)$. Mide la peor violación de tiempos de entrega (due dates).
- **Total weighted completion time** ($\sum w_j C_j$): La suma ponderada de los tiempos de completación de los n trabajos entregan una indicación del costo total de inventario

incurrido en la programación. La suma de los tiempos de completación es conocido generalmente en la literatura como flow time. Esta función objetivo es entonces conocida como weighted flow time.

- **Discounted total weighted completion time** ($\sum w_j(1 - e^{-rC_j})$): Esta es una función de costo más general que el anterior, donde los costos son descontados a un ratio de r , $0 < r < 1$, por unidad de tiempo. Esto es, si el trabajo j no es completado para el tiempo t se incurre en un costo adicional $w_j r e^{-rt} dt$ sobre el período $[t, t + dt]$. Si el trabajo j es completado para el tiempo t el costo total incurrido sobre el período $[0, t]$ es $w_j(1 - e^{-rt})$. El valor de r es usualmente cercano a cero, por ejemplo 0,1 o 10 %.
- **Total weighted tardiness** ($\sum w_j T_j$): Esto también es una función de costo más general que el de total weighted completion time.
- **Weighted number of tardy jobs** ($\sum w_j U_j$): Esta función objetivo no es sólo una medida de interés académico, es a menudo un objetivo en la práctica ya que es una medida que puede ser registrada muy fácilmente.

Como en el problema a abordar en este trabajo se desea mejorar el proceso de programación de la producción disminuyendo el tiempo total del proceso de los productos, la función objetivo del modelo propuesto corresponde a la minimización del **makespan**, el cual también ayuda a una mejor utilización de las máquinas.

2.1.2. Programación de la producción basado en el Problema del Vendedor Viajero (TSP)

Muchos de los trabajos presentes en la literatura que muestran un modelo de programación entera mixta incluyen un modelo basado en el Problema del Vendedor (TSP, por sus siglas en inglés Travelling Salesman Problem). Esto significa que se utiliza un modelo de ruteo para representar un problema de programación. Para aclarar esto, primero se detalla el TSP, el cual corresponde a lo siguiente: Dada una lista de ciudades y las distancias entre cada par de ellas, un vendedor, partiendo desde una ciudad de origen, debe visitar todas las ciudades exactamente una vez y regresar a la ciudad origen recorriendo la menor distancia

posible. En este problema se considera que la distancia entre dos ciudades es la misma en ambos sentidos, pero cuando la distancia desde la ciudad A hacia la ciudad B no es igual que la distancia desde B hacia A el problema es conocido como el Problema del Vendedor Viajero Asimétrico (ATSP), el cual será llamado en este trabajo desde ahora en adelante simplemente como TSP.

Para realizar la analogía de este problema con el problema de programación, se debe entender que cada ciudad corresponde a un trabajo a programar, las distancias entre ciudades corresponde al tiempo de setup entre los trabajos, los cuales son dependientes de la secuencia, el vendedor corresponde a la máquina que procesa los trabajos y la minimización de distancia recorrida corresponde a minimizar el tiempo total en procesar todos los trabajos. Dadas estas características básicas, el entorno considerado corresponde a una sola máquina con tiempos de setup dependientes de la secuencia y minimización del makespan. Si denotamos por (j, k) un par cualquiera de trabajos a programar, la clasificación de este problema utilizando la notación de tres campos corresponde a $1/s_{jk}/C_{\text{máx}}$.

Para entender mejor cómo se utiliza el TSP para modelar un problema de programación se presentará la formulación del TSP, utilizado como base para dar origen al modelo propuesto en este trabajo.

Formulación del TSP

En el grafo directo completo dirigido $D = (N, A)$, con arcos de distancias d_{jk} , se busca la ruta (un ciclo dirigido que contiene todas las n ciudades) de menor distancia. Se definen las variables

$$x_{jk} = \begin{cases} 1 & \text{si el arco } (j, k) \text{ está en la ruta,} \\ 0 & \text{en otro caso.} \end{cases}$$

El modelo de programación entera corresponde a:

$$\min \sum_{(j,k)} d_{jk} x_{jk} \quad (2.1)$$

$$s.a. \sum_j x_{jk} = 1 \quad \forall j \quad (2.2)$$

$$\sum_k x_{jk} = 1 \quad \forall j \quad (2.3)$$

$$x_{jk} \in \{0, 1\} \quad \forall j, k \quad (2.4)$$

La función objetivo (2.1) minimiza las distancia total de la ruta. Las restricciones (2.2) y (2.3) indican que sólo un arco debe seleccionarse para entrar y salir de algún nodo determinado, o bien que el vendedor debe entrar y salir sólo una vez de cada ciudad. La restricción (2.4) indica la naturaleza binaria de usar o no un arco determinado.

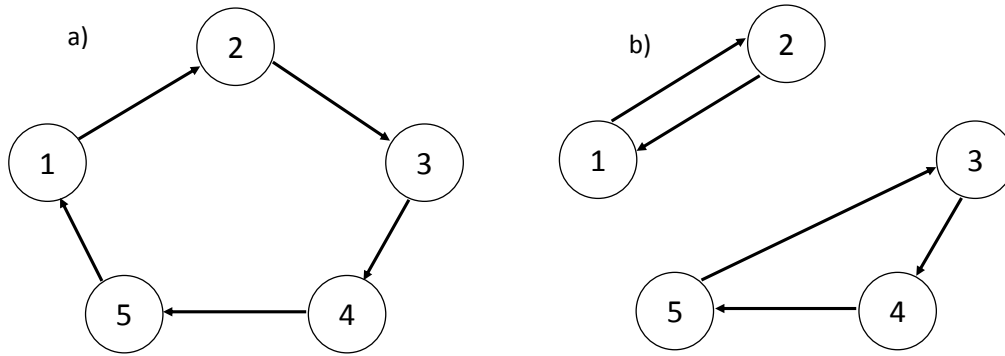


Figura 2.2: Posibles soluciones del TSP.

(Fuente: Elaboración propia.)

Esta formulación parece cumplir con todos los requisitos para resolver el problema, sin embargo la solución encontrada puede presentar un gran problema: la existencia de sub-rutas. En la Figura 2.2 se muestran los 2 tipos de soluciones que pueden ser encontradas al resolver este modelo para 5 ciudades (nodos). En esta figura, ambas soluciones establecen que a cada ciudad llega un arco determinado y sale un arco determinado. Sin embargo, la solución a) cumple con lo establecido en el problema del vendedor viajero, de comenzar en una ciudad, visitar solamente una vez todas las ciudades, y volver a la ciudad inicial. La solución b) presenta el llamado problema de sub-rutas que se podría originar al resolver el modelo.

Existen diversas formulaciones para eliminar las soluciones que contengan sub-rutas, correspondientes a la adición de nuevas variables y restricciones al modelo del TSP. Una de las formulaciones más conocidas corresponde a la de Miller–Tucker–Zemlin (MTZ), el cual asigna un número a cada ciudad de acuerdo al orden en que es visitada en la ruta del vendedor. La formulación incluye al modelo TSP una nueva variable $u_j (j = 1, \dots, n)$ y las siguientes restricciones:

$$u_1 = 1 \quad (2.5)$$

$$2 \leq u_j \leq n \quad \forall j \neq 1 \quad (2.6)$$

$$u_k + (n - 1)(1 - x_{jk}) \geq u_j + 1 \quad \forall j \neq 1, \forall k \neq 1 \quad (2.7)$$

La restricción (2.5) establece que se debe partir del nodo 1. La restricción (2.6) establece que el número de orden asignado al resto de las ciudades esté entre 2 y el número de ciudades. La restricción (2.7) asegura que una ciudad que es visitada después de otra, debe tener un número de orden mayor. Esta restricción sólo tiene sentido cuando no existen sub-rutas, es decir, una solución como la mostrada en a) en la Figura 2.2, ya que cuando $x_{jk} = 1$ la restricción queda $u_k \geq u_j + 1$, lo que significa que el siguiente nodo en ser visitado debe tener un número mayor hasta romper el ciclo en el nodo 1, ya que la restricción (2.7) no está definida para el nodo 1. Si existiesen sub-rutas significa que al menos una de ellas no contendrá el nodo 1, por lo que los valores de u_j incrementarían a infinito. La constante $(n - 1)$ sirve como una constante suficientemente grande para desactivar la restricción en caso que $x_{jk} = 0$. De esta manera, el modelo compuesto por (2.1) - (2.7) corresponde a la formulación MTZ del TSP.

2.1.3. Ejemplos de la literatura de modelos basados en el TSP

La formulación descrita en la sección anterior del TSP es utilizada para construir diferentes modelos y entornos de la programación de la producción. Un ejemplo de esto es el modelo propuesto en [Ríos-Mercado y Bard \(2003\)](#), donde se presenta un modelo de programación entera mixta para un entorno flow shop con tiempos dependientes de la

secuencia y minimización del makespan. Como en este entorno existen varias máquinas en serie, s_{ijk} corresponde al tiempo de setup dependiente de la secuencia del trabajo j al trabajo k en la máquina i , y la clasificación de este problema, asumiendo que el orden de los trabajos se mantiene en todas las máquinas, corresponde a $Fm/s_{ijk}, prmu/C_{\max}$. Para crear una programación en que se cumpla la situación del TSP en que se debe empezar y terminar en la misma ciudad, se debe crear un trabajo ficticio (dummy job) con tiempo de proceso nulo, el cual, al ser el primer trabajo a procesar, define un setup inicial de las máquinas al momento de programar el primer trabajo real. Como también se debe finalizar con este trabajo ficticio, este último tiempo de setup corresponde a limpieza de las máquinas luego de procesar el último trabajo real. En este entorno, un conjunto de n trabajos deben ser programados en un conjunto de m máquinas, donde cada trabajo tiene la misma secuencia de máquinas por las que ser procesado. Los detalles del modelo se presentan a continuación.

Conjuntos e índices:

- m Número de máquinas.
- n Número de trabajos.
- i Índice para las máquinas; $i \in I = 1, 2, \dots, m$.
- j, k Índice para los trabajos; $j, k \in J = 1, 2, \dots, n$.
- J_0 Corresponde a una extensión del conjunto de trabajos que incluye el trabajo ficticio denotado por 0; $J_0 = J \cup \{0\}$.

Parámetros:

- p_{ij} Tiempo de proceso del trabajo j en la máquina i ; $i \in I, j \in J$.
- s_{ijk} Tiempo de setup dependiente de la secuencia del trabajo j al trabajo k en la máquina i ; $i \in I, j \in J_0, k \in J$.
- A_i Constante que determina el límite superior del tiempo en que la máquina i termina de procesar su último trabajo, donde $A_0 = 0$; $i \in I$.
- B_{ij} Constante que determina el límite inferior del tiempo en que la máquina i comienza a procesar el trabajo j , donde $B_{0j} = 0 \forall j$; $i \in I, j \in J$.

Variables:

Sea $A = \{(j, k) : j, k \in J_0, j \neq k\}$ el conjunto de arcos en un grafo dirigido inducido por el nodo conjunto J_0 . Se definen las siguientes variables:

$$x_{jk} = \begin{cases} 1 & \text{si el trabajo } j \text{ es el inmediato antecesor del trabajo } k; (j, k) \in A, \\ 0 & \text{en otro caso,} \end{cases}$$

y_{ij} Variable real no negativa igual al tiempo en que comienza el procesamiento (starting time) del trabajo j en la máquina i ; $i \in I, j \in J$.

$C_{\text{máx}}$ Variable real no negativa igual al makespan.

$$\text{Minimizar } C_{\text{máx}} \quad (2.8)$$

$$\text{sujeto a } \sum_{j \in J_0} x_{jk} = 1 \quad k \in J_0 \quad (2.9)$$

$$\sum_{k \in J_0} x_{jk} = 1 \quad j \in J_0 \quad (2.10)$$

$$y_{ik} + A_i(1 - x_{jk}) \geq y_{ij} + p_{ij} + s_{ijk} \quad i \in I, j, k \in J \quad (2.11)$$

$$y_{mj} + p_{mj} \leq C_{\text{máx}} \quad j \in J \quad (2.12)$$

$$y_{ij} + p_{ij} \leq y_{i+1,j} \quad i \in I \setminus \{m\}, j \in J \quad (2.13)$$

$$x_{jk} \in \{0, 1\} \quad j, k \in J_0, j \neq k \quad (2.14)$$

$$y_{ij} \geq B_{ij} \quad i \in I, j \in J \quad (2.15)$$

La función objetivo del makespan es minimizada en (2.8). Las ecuaciones (2.9) y (2.10) son iguales a las ecuaciones (2.2) y (2.3) utilizadas por el TSP. En este caso determinan que cada trabajo debe tener sólo un antecesor y sólo un sucesor, respectivamente. La ecuación (2.11) es la restricción de eliminación de sub-rutas basada en la restricción (2.7) de la formulación MTZ. Establece que si el trabajo j antecede al trabajo k , entonces el tiempo en que comienza el trabajo k en la máquina i debe ser mayor al tiempo de completación del trabajo j en la máquina i ($y_{ij} + p_{ij}$) más el tiempo de setup correspondiente. Aquí, A_i es un número suficientemente grande (un límite superior para el tiempo de completación en la máquina i). La restricción (2.12) determina el makespan, asegurando que sea mayor o igual al tiempo de completación de todos los trabajos en la última máquina. La restricción

(2.13) establece que un trabajo no puede empezar su procesamiento en una máquina si no ha terminado su proceso en la máquina anterior. Finalmente, en la restricción (2.14) se define la naturaleza binaria de la variable x_{jk} , y en la restricción (2.15) se establece un límite inferior para el comienzo del procesamiento de cada trabajo en cada máquina.

Otro ejemplo de un modelo de programación basado en el TSP un poco más complejo es el que se presenta en Radhakrishnan y Ventura (2000), donde se modela el entorno $P/s_{jk}/\sum E_j + \sum T_j$. En este entorno, se deben programar N trabajos con tiempos de setup dependientes de la secuencia, tiempos de entrega (due dates) y tiempos de procesos variables en M máquinas idénticas en paralelo. Los detalles del modelo se presentan a continuación.

Conjuntos e índices:

- M Número de máquinas.
- N Número de trabajos.
- i Índice para las máquinas; $i \in I = 1, 2, \dots, M$.
- j, k Índice para los trabajos; $j, k \in J = 1, 2, \dots, N$.

Parámetros:

- d_j Tiempo de entrega del trabajo j .
- p_j Tiempo de proceso del trabajo j .
- s_{jk} tiempo de setup del trabajo j al trabajo k .
- W Un número suficientemente grande.

Variables:

- c_j Variable real que representa el tiempo de completación del trabajo j .
- e_j Variable real que representa el earliness del trabajo j . Esta variables es similar al concepto de tardiness (sección 2.1); ambas son siempre positivas pero earliness

mide el tiempo de holgura cuando el trabajo j es completado antes de su tiempo de entrega.

t_j Variable real que representa el tardiness del trabajo j .

$$x_{ijk} = \begin{cases} 1 & \text{si el trabajo } j \text{ es el inmediato antecesor del trabajo } k \text{ en la máquina } i, \\ 0 & \text{en otro caso,} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si el trabajo } j \text{ es asignado a la máquina } i, \\ 0 & \text{en otro caso.} \end{cases}$$

$$\text{Minimizar} \quad \sum_{j=1}^N (e_j + t_j) \quad (2.16)$$

$$\text{sujeto a} \quad \sum_{k=1}^N x_{i0k} = 1 \quad i = 1, \dots, M \quad (2.17)$$

$$\sum_{j=0, j \neq k}^N x_{ijk} = y_{ik} \quad k = 1, \dots, N, i = 1, \dots, M \quad (2.18)$$

$$\sum_{k=0, k \neq j}^N x_{ijk} = y_{ij} \quad j = 1, \dots, N, i = 1, \dots, M \quad (2.19)$$

$$c_k + W(1 - x_{i0k}) \geq s_{0k} + p_k \quad k = 1, \dots, N, i = 1, \dots, M \quad (2.20)$$

$$c_k + W(1 - x_{ijk}) \geq c_j + s_{jk} + p_k \quad \begin{matrix} j = 1, \dots, N, k = 1, \dots, N, \\ i = 1, \dots, M, j \neq k \end{matrix} \quad (2.21)$$

$$c_j - d_j = t_j - e_j \quad j = 1, \dots, N \quad (2.22)$$

$$\sum_{i=1}^M y_{ij} = 1 \quad j = 1, \dots, N \quad (2.23)$$

$$c_j, t_j, e_j \geq 0 \quad j = 1, \dots, N \quad (2.24)$$

$$x_{ijk}, y_{ij} \in \{0, 1\} \quad \begin{matrix} j = 1, \dots, N, k = 1, \dots, N, \\ i = 1, \dots, M, j \neq k \end{matrix} \quad (2.25)$$

La función objetivo (2.16) corresponde a la minimización de la suma del earliness y tardiness para los N trabajos. Esto implica que se desea buscar una programación en que los productos sean completados lo más cerca de su tiempo de entrega. En la restricción (2.17) se asigna el trabajo ficticio 0 al principio de la secuencia antes de todos los trabajos reales

en cada máquina. Las restricciones (2.18) y (2.19) son similares a las restricciones (2.2) y (2.3) del TSP, pero en vez de utilizar la constante 1 en el lado derecho, se utiliza la variable binaria y_{ij} , ya que al tener máquinas en paralelo la secuencia debe existir sólo si ambos trabajos fueron asignados a esa máquina. Las restricciones (2.20) y (2.21) corresponden a restricciones de eliminación de sub-rutas basadas en la restricción (2.7) de la formulación MTZ y cumplen una función similar a la restricción (2.11) del ejemplo anterior, con la diferencia que ahora las posiciones de los trabajos en cada máquina son ordenados por su tiempo de completación c_j . En este ejemplo los autores decidieron dividir esta restricción en dos, la primera (2.20) para computar el tiempo de completación del primer trabajo en cada máquina y la segunda (2.21) para computar el tiempo de completación del resto de los trabajos. La restricción (2.22) relaciona los tiempos de completación de cada trabajo real con su tiempo de entrega y las variables de earliness y tardiness. La restricción (2.23) define que un trabajo real debe ser asignado a sólo una máquina. Finalmente, las restricciones (2.24) y (2.25) definen el dominio de las variables.

Los dos ejemplos presentados en esta sección muestran un modelo de programación entera mixta basado en el TSP y tienen componentes que son incluidos en el modelo propuesto en este trabajo. El objetivo de mostrar estos modelos es introducir al lector a entender la base que dan origen al modelo propuesto en la sección 3.2.

2.2. Estado del Arte

En esta sección se describen brevemente otros artículos relacionados con el problema abordado en este trabajo. En Pinedo (2015) se tiene una introducción a la teoría, algoritmos y sistemas de la programación en general. En este libro se presentan los antecedentes, tipos de programación, aplicaciones y también se plantean las dificultades computacionales de estos problemas, que debido a sus características combinatoriales son clasificados en general como NP-Hard.

Con respecto a la programación de la producción, la inclusión de setup dependientes de

la secuencia es un tema abundantemente investigado, véase [Allahverdi et al. \(2008\)](#), donde se presenta una revisión de la literatura que aborda este problema. En [Ríos-Mercado y Bard \(1998\)](#) se presentan dos modelos de programación lineal entera (MIP) con minimización del makespan para un ambiente de Flow Shop con tiempos de setup dependientes de la secuencia, que luego son resueltos por algoritmos de Branch-and-Cut. Estos modelos son utilizados en [Kurz y Askin \(2004\)](#) y [Karmakar y Mahanty \(2010\)](#), donde son extendidos a un ambiente de Flexible Flow Lines, con máquinas idénticas en paralelo en cada etapa. En [Rocha et al. \(2008\)](#) se presentan dos modelos MIP para la programación en máquinas no relacionadas en paralelo, considerando tiempos de setup dependientes de la secuencia. En [Ruiz y Andrés-Romano \(2011\)](#) se resuelve un modelo MIP de programación de la producción en máquinas no relacionadas en paralelo con asignación de recursos, donde los tiempos de setup no sólo dependen de la secuencia, sino que también de los recursos que se le asignan, como la cantidad de personas disponibles. En [Yilmaz Eroglu y Ozmutlu \(2014\)](#) se presentan modelos MIP para la programación en máquinas no relacionadas en paralelo junto con la propiedad de job-splitting, donde un job puede ser dividido entre las máquinas disponibles para ser procesado separadamente. La resolución de estos modelos es a través de heurísticas híbridas entre los métodos de Algoritmos Genéticos y Búsqueda Local. En [Chen et al. \(2006\)](#) se resuelve el problema de sólo una máquina con setup dependientes de la secuencia, junto con la propiedad de batching, donde los jobs con setup similares son agrupados en batches y procesados en conjunto, el problema es resuelto por Algoritmo Genético. [Hnaïen et al. \(2015\)](#) aborda un ambiente Flow Shop de dos máquinas con restricciones de disponibilidad en la primera máquina. En él se presentan dos modelos MIP y un algoritmo de Branch and Bound (B&B) basado en un conjunto de nuevas cotas inferiores, además de otras heurísticas. En [Jia et al. \(2015\)](#) se resuelve la minimización del tiempo de producción (Makespan) de lotes en máquinas en paralelo con distintas capacidades, a través de dos heurísticas diferentes. La primera es basada en la regla de First-Fit-Decreasing (FFD) y la segunda en Max-Min Ant System (MMAS).

Todos los trabajos mencionados anteriormente tienen modelos de tiempo continuo y la misma función objetivo: minimización del tiempo de producción (Makespan minimization).

En relación con el presente trabajo, el método de resolución también es a través de un modelo de programación entera mixta (MIP), pero al ser aplicado a problemas en la industria de bebidas no alcohólicas surgen desafíos adicionales, tales como la sincronización entre las etapas de producción, lo que será explicado en la sección 3.1.

La planificación en conjunto con la programación de la producción también entrega soluciones para el problema en cuestión, donde existe un dimensionamiento de lote simultáneamente con la programación de la producción. Para mayor información, véase [Copil et al. \(2016\)](#), el cual presenta un survey con respecto a este tema y su revisión de la literatura. Uno de los trabajos importantes en este tema corresponde a [Fleischmann y Meyr \(1997\)](#), donde se presenta un modelo de planificación y programación con setup dependientes de la secuencia que minimiza los costos de setup e inventario, llamado The General Lot-Sizing and Scheduling Problem (GLSP). En este artículo, la técnica para determinar simultáneamente el lote y su programación corresponde a la utilización de una estructura especial del tiempo, el cual es dividido en macro periodos, en donde, cada macro periodo es subdividido en micro periodos. En cada macro periodo se ven envueltos los elementos que proporcionan la información externa al problema, como la demanda y los costos de inventario, y los micro periodos determinan el orden en que se producen los productos, ya que en cada micro periodo se permite la producción de sólo un producto. En [Meyr \(2000\)](#) se extiende el GLSP al incluirle tiempos de setup dependientes de la secuencia, llamado GLSPST, donde el método de resolución del modelo corresponde a un algoritmo de re-optimización dual combinado con heurísticas de Búsqueda Local. En [Meyr \(2002\)](#) el GLSPST anterior es extendido a líneas de producción en paralelo, llamado GLSPPL, el cual es resuelto de manera similar. En [Seeanner y Meyr \(2013\)](#) se realiza una nueva extensión, esta vez a un ambiente multi etapa y, adicionalmente, se incorporan propiedades que permiten utilizar de mejor manera las líneas de producción: quantity and setup splitting, que permiten dividir cantidades y tiempos de setup en micro periodos consecutivos. Se generan instancias pequeñas que son resueltas por solver estándar de programación entera mixta y heurísticas de Relax-and-Fix, para luego ser comparados.

En [Ferreira et al. \(2009\)](#) se utiliza un modelo basado en el modelo GLSP, pero aplicado a la industria de bebidas no alcohólicas, en el cual se presenta un modelo de dos etapas con máquinas en paralelo, tiempos dependientes de la secuencia y la sincronización entre ambas etapas, lo que se ajusta bastante bien a la problemática a resolver en este trabajo, exceptuando la diferencia en que el objetivo del GLSP es un plan a largo plazo con minimización de los costos de inventario y setup. El modelo es resuelto mediante una aproximación relajada y heurísticas del tipo Relax-and-Fix. Este mismo modelo es resuelto en [Toledo et al. \(2011\)](#), pero utilizando algoritmos de Búsqueda Tabú. En [Ferreira et al. \(2012\)](#) se presentan 4 formulaciones de sólo una etapa para modelar la problemática de dos etapas de la industria de bebidas no alcohólicas abordada en los artículos anteriores, dos de ellas en base al GLSP y las otras dos en base al Problema del Vendedor Viajero Asimétrico (ATSP). En [Toledo et al. \(2014\)](#) también se resuelve el modelo presentado en [Ferreira et al. \(2009\)](#), pero en este caso se utiliza la combinación de Algoritmo Genético con técnicas de programación matemática.

En estos trabajos relacionados a la industria de bebidas no alcohólicas se presentan modelos de dimensionamiento de lote en conjunto con la programación de la producción, con el objetivo de minimizar los costos en el largo plazo, lo que se diferencia del modelo propuesto en este trabajo. El principal objetivo del presente trabajo es proveer una programación de la producción a corto plazo con minimización del tiempo total de producción. Otra diferencia con los artículos anteriores es que se permite la división de un lote para ser programado en diferentes tanques, lo que reduce tiempos de espera en las líneas de envasado. Además de esto, se incorpora una estrategia de resolución utilizando el plan actual de la empresa como parte de la resolución del modelo, la cual provee una cota superior que representa el tiempo total de proceso de los productos a programar, a partir de la cual se busca minimizar.

3 | Metodología

En este capítulo se presenta en detalle el problema a resolver, describiendo las principales características del proceso productivo. También se presenta el modelo de programación lineal entera mixta que representa la problemática a resolver, para luego mostrar la metodología de resolución, utilizando heurísticas para la reducción del problema. Esta metodología fue presentada y publicada en [Leiva y Alborno \(2016\)](#).

3.1. Descripción del problema: El proceso productivo

El proceso productivo de la industria de bebidas no alcohólicas es similar a un entorno de Flexible Flow Shop, con ciertas características únicas de esta industria, como lo es la sincronización de las etapas productivas, explicado más adelante.

La producción de bebidas no alcohólicas es realizada principalmente en dos etapas interdependientes claramente identificables. La primera etapa de producción corresponde a la preparación del jarabe, donde se producen los diferentes tipos de bebidas, ya sean jugos, aguas, bebidas carbonatadas o bebidas funcionales. La preparación del jarabe tiene lugar en los tanques de acumulación, donde se mezclan los ingredientes y se realizan los controles de calidad correspondientes, cuidando las proporciones de los componentes e inocuidad del jarabe. Luego que el jarabe está listo, se procede a la segunda etapa de producción, correspondiente al embotellado del jarabe en las líneas de envasado. En esta etapa del proceso, se transfiere el jarabe desde los tanques de acumulación hacia las líneas de envasado, específicamente hacia la llenadora, donde el jarabe es embotellado, inspeccionado, codificado, etiquetado, encajonado y paletizado. Todos estos procesos son realizados

automáticamente en serie en un proceso continuo mediante cintas transportadoras, desde la acumulación de botellas vacías hasta la creación de pallets de productos terminados. Esta serie de máquinas corresponden a la línea de envasado. Debido a estas características, y sin pérdida de generalidad, las líneas de envasado serán consideradas en este trabajo como una sola máquina.

En la primera etapa de producción se cuenta con varios tanques de acumulación de diferentes capacidades y cada tanque puede almacenar un subconjunto de los sabores que produce la planta. En la segunda etapa de producción, también se cuenta con varias líneas de envasado, con diferentes velocidades de producción y cada línea puede envasar un subconjunto de los formatos que produce la planta. Para surtir el jarabe hacia las líneas de envasado, los tanques pueden ser conectados a cualquier línea. En el caso de las líneas, pueden recibir jarabe desde cualquier tanque, pero sólo desde un tanque a la vez, es decir, no puede haber más de un tanque conectado a una línea al mismo tiempo. En la Figura 3.1 se muestra una representación esquemática del proceso productivo.

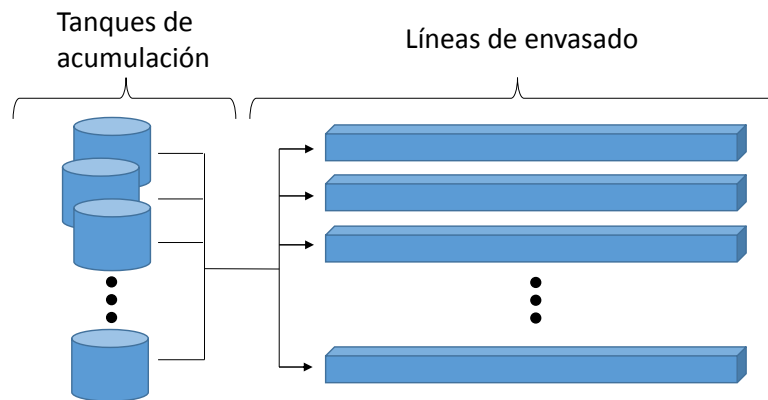


Figura 3.1: Proceso productivo de la industria de bebidas no alcohólicas.

(Fuente: Elaboración propia.)

Como se mencionó anteriormente, durante el proceso de producción existen tiempos de setup dependientes de la secuencia. En la primera etapa de producción, los tiempos de setup corresponden a la limpieza del tanque más la preparación del nuevo jarabe en ese mismo tanque. Estos tiempos de limpieza y preparación son dependientes de la secuencia debido a las características del jarabe a preparar. Las bebidas light, jugos néctar

y bebidas funcionales requieren de estrictos procedimientos para su preparación, por lo que el tiempo de limpieza y preparación desde una bebida normal a una light no es igual a realizar el proceso inverso. En la segunda etapa de producción también existen tiempos de setup dependientes de la secuencia, debido a las características de los formatos de los envases utilizados, que pueden ser de plástico, vidrio, lata, y de diferentes tamaños cada uno.

Aunque existen tiempos de setup dependientes de la secuencia en ambas etapas productivas, los de la primera etapa son considerablemente mayores, los que pueden variar entre 2 y 12 horas. Además de esto, existen tipos de bebidas en que los lotes de un producto superan la capacidad del tanque disponible, por lo que se requiere que la línea que está envasando deba esperar la preparación de más jarabe luego que se vacía ese tanque. Es por esto que, en general, las plantas embotelladoras cuentan con un número de tanques mayor que el número de líneas de envasado. Esta característica es tomada en cuenta en la formulación del modelo de la sección 3.2, ya que se permite la división del lote de un producto para ser acumulado en más de un tanque, lo que permite surtir parte del lote desde un tanque mientras se prepara el resto del lote en otros tanques. De esta manera, las líneas de envasado se mantienen mayor tiempo en operación y disminuyen sus tiempos de espera.

A pesar de existir dos etapas productivas diferentes, la producción se lleva a cabo simultáneamente en ambas etapas, por lo que se debe considerar la sincronización entre ellas, ya que una línea de envasado no puede operar sin que el tanque esté listo para proveer el líquido correspondiente, y, de la misma forma, un tanque no puede surtir el jarabe sin que la línea esté lista para envasar. En la Figura 3.2 se muestra un ejemplo donde se puede apreciar la importancia de la sincronización entre las etapas, ya que una programación no sincronizada conduce a una solución infactible. Esta característica es particular en la industria de bebidas no alcohólicas, y establece una diferencia clave con un ambiente tipo Flow Shop, donde los productos son procesados secuencialmente en cada etapa. En la Figura 3.2 también se puede apreciar la división del lote en más de un tanque, el cual debe estar sincronizado con la utilización de la línea que procesa ese lote, teniendo cuidado de que no se traslapen los lotes entre tanques, ya que, como se mencionó anteriormente, no

puede haber más de un tanque surtiendo jarabe a una línea al mismo tiempo. Para lograr esto, el modelo propuesto asigna una posición a cada tanque en que el lote del producto fue dividido, indexado por el subíndice “o”. De esta manera, el orden en que los tanques proveerán jarabe a la línea de envasado está dado por las posiciones asignadas.

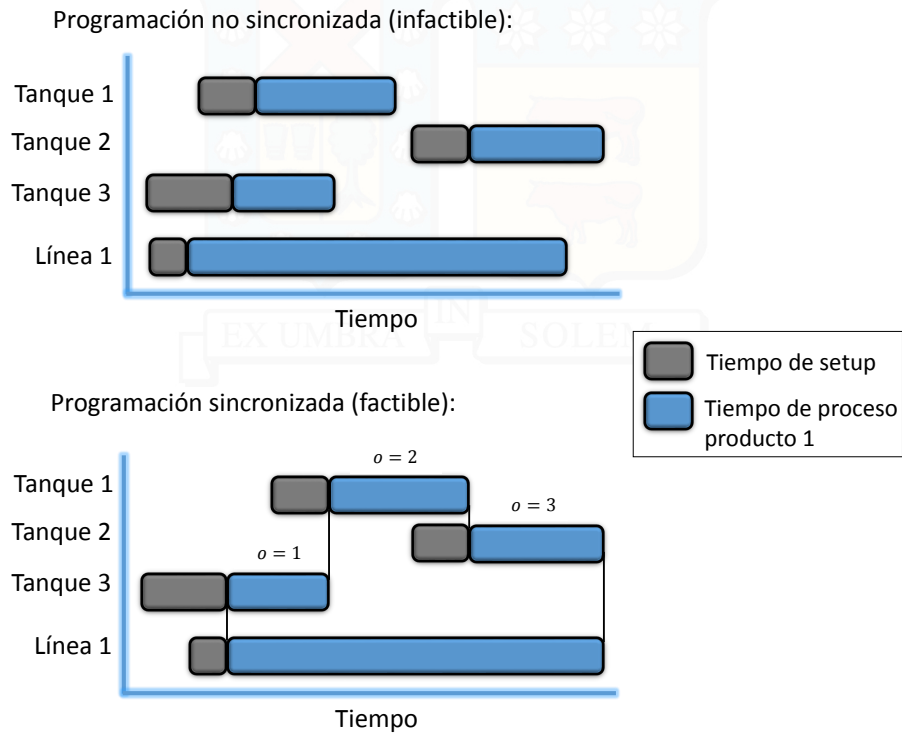


Figura 3.2: Sincronización entre las etapas productivas.

(Fuente: Elaboración propia.)

Los productos a programar son identificados por dos características principales, el sabor del jarabe y el formato de embotellado. La identificación de cada producto viene dada por su SKU (Stock Keeping Unit), que corresponde a un número identificador con el objeto de permitir el seguimiento sistémico de los productos. Para determinar la cantidad a producir de cada producto, la planta genera un plan de producción en cierto período de tiempo.

En el caso aplicado en este trabajo, la firma cuenta con 104 sabores y 51 formatos distintos, con un total de aproximadamente 280 SKU's diferentes. El plan de producción se realiza para una semana y en cada semana se programan entre 100 y 150 del total de productos. Los productos son elaborados en 15 líneas diferentes de producción y 36 tanques de acumulación con capacidades que van desde los 12 mil litros a los 40 mil litros. La firma

también cuenta con indicadores de desempeño como el cumplimiento de los Días Venta, que corresponde al tiempo de entrega que tienen algunos de los productos a programar. Este indicador debe ser cumplido estrictamente, es decir, no puede ser violado en ningún caso, por lo que corresponde a restricción de deadline (2.1). El objetivo del modelo propuesto es la minimización del tiempo de producción, por lo que si se logra disminuir el tiempo del plan actual, la planta podrá planificar mayor cantidad de productos en el mismo período de tiempo, aumentando su eficiencia.

3.2. Modelo propuesto

A continuación se detalla el modelo propuesto y sus características. Es importante mencionar primero las consideraciones tomadas para la creación de este modelo. Los lotes de productos a programar vienen identificados por su SKU, por lo que, cuando un producto es asignado a un tanque significa que en ese tanque sólo se almacena el jarabe que será destinado a ese producto, determinado por su SKU. Se asume que siempre existe disponibilidad de la materia prima necesaria para la preparación de jarabe. Un producto puede ser asignado a más de un tanque, pero puede ser asignado a sólo una línea, por lo que es conveniente maximizar su utilización con la división de lotes. La demanda de un producto corresponde al plan generado por la empresa. Un lote corresponde a la cantidad demandada del producto. Los parámetros son parecidos a los expuestos en [Ferreira et al. \(2009\)](#), ya que exponen las características de la industria aquí estudiada.

Las variables del modelo están referidas a la etapa productiva en la que actúan, el superíndice “ T ” se refiere a la primera etapa: Tanques, el superíndice “ L ” se refiere a la segunda etapa: Líneas. Las restricciones están divididas en tres partes, basados en la etapa en la que actúan y la sincronización.

Conjuntos:

J Conjunto de productos.

J_0 Conjunto de productos incluyendo un producto ficticio 0 (define setup inicial y final)

$$(J_0 = J \cup \{0\}).$$

α_m Conjunto de productos que pueden ser producidos en el tanque/línea m .

β_j Conjunto de tanques que pueden producir el producto j .

γ_j Conjunto de líneas que pueden producir el producto j .

Parámetros:

D_j Demanda del producto j .

d_j Tiempo de entrega (deadline) del producto j .

s_{ij}^T Tiempo de set up del producto i al j en Tanques.

s_{ij}^L Tiempo de set up del producto i al j en Líneas.

a_{jm} Tiempo de proceso de una unidad del producto j en Línea m .

r_j Cantidad de líquido necesario para producir una unidad del producto j .

K_m Capacidad del Tanque m .

G Límite superior para el tiempo de completación de un producto.

Variables:

x_{ijm}^T 1 si el producto j es procesado inmediatamente después del producto i en el Tanque m ; 0 en otro caso.

x_{ijm}^L 1 si el producto j es procesado inmediatamente después del producto i en la Línea m ; 0 en otro caso.

y_{jmo}^T 1 si el producto j es procesado en el Tanque m en la posición o ; 0 en otro caso.

y_{jm}^L 1 si el producto j es procesado en la Línea m ; 0 en otro caso.

p_{jmo}^T Cantidad de líquido del producto j en el Tanque m en la posición o .

C_{jmo}^T Tiempo de completación del producto j en el Tanque m en la posición o .

C_j^L Tiempo de completación del producto j en las Líneas.

$C_{\max}$ Makespan ($C_{\max} = \max\{C_j^L\}$).

v_j Tiempo de espera de la línea mientras es procesado el producto j .

q_{jmo}^T Número de set up necesarios desde el producto j al j en el Tanque m en la posición o .

Función Objetivo:

$$\text{Min } C_{\text{máx}} \quad (3.1)$$

La función objetivo (3.1) corresponde a la minimización del máximo de los tiempos de completación de los productos, lo que reduce el tiempo total en que los productos permanecen en el sistema, determinando así el tiempo en que el último producto es completado.

Restricciones:■ **Programación de las operaciones en Tanques:**

$$\sum_{m \in \beta_j} \sum_{o=1}^{|\beta_j|} y_{jmo}^T \geq 1 \quad j \in J \quad (3.2)$$

La restricción (3.2) permite garantizar la asignación de un producto en al menos un tanque. El subíndice “o” indica la posición en que el tanque será usado por el producto j . Como existe una relación uno a uno entre los tanques y sus posiciones, es decir, cada tanque puede tener sólo una posición y cada posición es usado por sólo un tanque (restricciones (3.11) y (3.12)), la suma en “o” no afecta el resultado de las restricciones que no son designadas para la correcta asignación de posiciones entre los tanques.

$$\sum_{o=1}^{|\beta_0|} y_{0mo}^T = 1 \quad m \in \beta_0 \quad (3.3)$$

$$\sum_{j \in \alpha_m} x_{ijm}^T = \sum_{o=1}^{|\beta_i|} y_{imo}^T \quad i \in J_0, m \in \beta_i \quad (3.4)$$

$$\sum_{i \in \alpha_m} x_{ijm}^T = \sum_{o=1}^{|\beta_j|} y_{jmo}^T \quad j \in J_0, m \in \beta_j \quad (3.5)$$

Las restricciones (3.3), (3.4) y (3.5) corresponden a la correcta asignación y secuenciamiento de los productos en los tanques, usando la estructura del Problema del Vendedor Viajero (TSP) (ver sección 2.1.2). La primera restricción (3.3) asigna un producto ficticio 0 a todos los tanques, la segunda restricción (3.4) define que cada producto tenga sólo un sucesor en cada tanque siempre y cuando ese producto haya sido asignado a ese tanque, y la tercera restricción (3.5) define que cada producto tenga sólo un antecesor en cada tanque siempre y cuando ese producto haya sido asignado a ese tanque. El producto ficticio

es usado para crear una secuencia en que el primer y último producto en ser procesado sea siempre este producto ficticio, de manera que las restricciones (3.4) y (3.5) no sean violadas.

$$\sum_{o=1}^{|\beta_j|} C_{jmo}^T + G(2 - x_{ijm}^T - y_{jw}^L) \geq \sum_{o=1}^{|\beta_i|} C_{imo}^T + \sum_{o=1}^{|\beta_j|} s_{ij}^T y_{jmo}^T + \sum_{o=1}^{|\beta_j|} \frac{a_{jw} p_{jmo}^T}{r_j} + \sum_{o=1}^{|\beta_j|} q_{jmo}^T s_{jj}^T$$

$$i \in J_0, j \in J, w \in \gamma_j, m \in \beta_i \cap \beta_j \quad (3.6)$$

La restricción (3.6) define la correcta relación entre los tiempos de completación de cada producto en cada tanque. La diferencia entre los tiempos de completación de dos productos consecutivos está dada por los tres últimos términos de esta restricción: el tiempo de set up del producto i al j , el tiempo de proceso del producto j en el tanque m y los posibles tiempos de set up de rellenos de ese tanque si son necesarios. Esto último hace posible utilizar un tanque más de una vez para cada producto sin tener que asignarle una nueva posición “ o ”, ya que como los rellenos se realizan consecutivamente, se asigna sólo una posición a todos los posibles rellenos del tanque. De esta manera, q_{jmo}^T indica el número de tiempos de set up añadidos para llenar el tanque con líquido del mismo producto. Esta restricción sólo se activa cuando el producto j es procesado inmediatamente después del producto i en el tanque m ($x_{ijm}^T = 1$) y el producto j es asignado a la línea w ($y_{jw}^L = 1$), ya que el tiempo de proceso depende de la velocidad de la línea que procesa ese producto. El parámetro G corresponde a una constante lo suficientemente grande, el cual determina un límite superior para el tiempo de completación de un producto. Esta constante puede ser calculada de la siguiente manera:

$$G = \sum_{j \in J_0} \left(\max_{i \in J_0} \{s_{ij}^T\} + \left(\max_{w \in \gamma_j} \{a_{jw}\} \right) D_j + \left\lceil \frac{r_j D_j}{\min_{m \in \beta_j} \{K_m\}} \right\rceil s_{jj}^T \right) \quad (3.7)$$

Con esta ecuación es posible determinar este límite utilizando la peor programación posible, es decir, el tiempo en que se completarían todos los productos programados en la líneas más lenta, utilizando el tanque más pequeño y suponiendo el mayor de los tiempos de setup en tanques entre los productos, ya que son los que más impactan al tiempo total de

producción.

$$\sum_{m \in \beta_j} \sum_{o=1}^{|\beta_j|} p_{jmo}^T \geq r_j D_j \quad j \in J \quad (3.8)$$

$$p_{jmo}^T \leq r_j D_j \cdot y_{jmo}^T \quad j \in J, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.9)$$

$$C_{jmo}^T \leq G \cdot y_{jmo}^T \quad j \in J, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.10)$$

La restricción (3.8) permite cumplir con la demanda en litros de cada producto. Las restricciones (3.9) y (3.10) definen el límite superior para la cantidad de líquido y el tiempo de completación para cada producto en cada tanque, respectivamente.

$$\sum_{o=1}^{|\beta_j|} y_{jmo}^T \leq 1 \quad j \in J, m \in \beta_j \quad (3.11)$$

$$\sum_{m \in \beta_j} y_{jmo}^T \leq 1 \quad j \in J, o = 1 \dots |\beta_j| \quad (3.12)$$

$$\sum_{m \in \beta_j} y_{jm(o-1)}^T \geq \sum_{m \in \beta_j} y_{jmo}^T \quad j \in J, o = 2 \dots |\beta_j| \quad (3.13)$$

Como se mencionó anteriormente, las restricciones (3.11) y (3.12) definen la relación uno a uno entre los tanques y sus posiciones, es decir, en (3.11) un producto asignado a un tanque sólo puede tener una posición, y en (3.12) un producto con una posición debe usar sólo un tanque. La restricción (3.13) determina un orden coherente de las posiciones de los tanques para cada producto, esto es, si una posición no fue asignada, todas las siguientes posiciones no son asignadas.

$$q_{jmo}^T \leq \frac{p_{jmo}^T}{K_m} \quad j \in J, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.14)$$

$$q_{jmo}^T \geq \frac{p_{jmo}^T}{K_m} - 1 \quad j \in J, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.15)$$

$$x_{iim}^T = 0 \quad i \in J, m \in \beta_i \quad (3.16)$$

Las restricciones (3.14) y (3.15) determinan la máxima y mínima cantidad de rellenos que puede tener un producto en un tanque y la restricción (3.16) previene la programación de un mismo producto sucesivamente.

■ Programación de las operaciones en las Líneas:

$$\sum_{m \in \gamma_j} y_{jm}^L = 1 \quad j \in J \quad (3.17)$$

Con respecto a las restricciones en las líneas, la restricción (3.17) asigna productos a sólo una línea.

$$y_{0m}^L = 1 \quad m \in \gamma_0 \quad (3.18)$$

$$\sum_{j \in \alpha_m} x_{ijm}^L = y_{im}^L \quad i \in J_0, m \in \gamma_i \quad (3.19)$$

$$\sum_{i \in \alpha_m} x_{ijm}^L = y_{jm}^L \quad j \in J_0, m \in \gamma_j \quad (3.20)$$

Las restricciones (3.18), (3.19) y (3.20) corresponden a la correcta asignación y secuenciamiento de los productos en las líneas, usando la estructura del Problema del Vendedor Viajero (TSP) (ver sección 2.1.2) y son similares a las restricciones (3.3), (3.4) y (3.5).

$$C_j^L + G(1 - x_{ijm}^L) \geq C_i^L + s_{ij}^L + a_{jm}D_j + v_j \quad i \in J_0, j \in J, m \in \gamma_i \cap \gamma_j \quad (3.21)$$

La restricción (3.21) define la correcta relación entre los tiempos de completación de cada producto en cada línea. La diferencia entre los tiempos de completación de dos productos consecutivos está dada por los tres últimos términos de esta restricción: el tiempo de set up del producto i al j , el tiempo de proceso del producto j en la línea m y posibles tiempos de espera de la línea mientras se rellena el tanque. Es similar a la restricción (3.6).

$$C_{\max} \geq C_j^L \quad j \in J_0 \quad (3.22)$$

$$d_j \geq C_j^L \quad j \in J_0 \quad (3.23)$$

$$x_{iim}^L = 0 \quad i \in J, m \in \gamma_i \quad (3.24)$$

La restricción (3.22) determina el Makespan como el máximo de los tiempos de completación en las líneas. La restricción (3.23) permite el cumplimiento de los deadlines y la restricción (3.24) es similar a la restricción (3.16).

■ **Sincronización entre las etapas:**

$$C_j^L - a_{jw}D_j - v_j \leq G(1 - y_{jw}^L) + \sum_{m \in \beta_j} C_{jm1}^T - \sum_{m \in \beta_j} \frac{a_{jw}p_{jm1}^T}{r_j} - \sum_{m \in \beta_j} q_{jm1}^T s_{jj}^T \quad j \in J, w \in \gamma_j \quad (3.25)$$

Con respecto a las restricciones de sincronización, la restricción (3.25) define que el procesamiento de un producto en los tanques (tanque asignado en primera posición) no empiece antes que su procesamiento en las líneas, es decir, que el límite izquierdo del procesamiento de un producto en tanques y líneas de la Figura 3.2 coincidan.

$$\sum_{m \in \beta_j} C_{jmo}^T + G \left(2 - \sum_{m \in \beta_j} y_{jmo}^T - y_{jw}^L \right) \geq \sum_{m \in \beta_j} C_{jm(o-1)}^T + \sum_{m \in \beta_j} \frac{a_{jw}p_{jmo}^T}{r_j} + \sum_{m \in \beta_j} q_{jmo}^T s_{jj}^T \quad j \in J, o = 2 \dots |\beta_j|, w \in \gamma_j \quad (3.26)$$

La restricción (3.26) define el correcto secuenciamiento de las posiciones para los tanques, evitando que los tiempos de procesos en los tanques se traslapen en la producción de un producto.

$$C_j^L \geq \sum_{o=1}^{|\beta_j|} C_{jmo}^T \quad j \in J, m \in \beta_j \quad (3.27)$$

La restricción (3.27) permite que el procesamiento de un producto en los tanques no exceda su procesamiento en la línea, es decir, que el límite derecho del procesamiento de un producto en tanques y líneas de la Figura 3.2 coincidan.

$$x_{ijm}^T, y_{jmo}^T \in \{0, 1\} \quad i, j \in J_0, m \in \beta_i \cap \beta_j, o = 1 \dots |\beta_j| \quad (3.28)$$

$$x_{ijm}^L, y_{jm}^L \in \{0, 1\} \quad i, j \in J_0, m \in \gamma_i \cap \gamma_j \quad (3.29)$$

$$q_{jmo}^T \in Z_0^+ \quad j \in J, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.30)$$

$$C_j^L, C_{jmo}^T, C_{\text{máx}}, p_{jmo}^T, v_j \geq 0 \quad j \in J_0, m \in \beta_j, o = 1 \dots |\beta_j| \quad (3.31)$$

Finalmente, las restricciones (3.28), (3.29), (3.30) and (3.31) definen el dominio de las variables de decisión.

3.3. Estrategia de aplicación del modelo

Debido a la complejidad del modelo y su resolución para instancias grandes, se utiliza una metodología especial de resolución, basado en heurísticas para la reducción del problema. Esta metodología incluye dos fases principales: División del problema en clusters y programación a priori según criterios de decisión.

3.3.1. División del problema en clusters

En esta primera fase de la estrategia, el problema es dividido en clusters, lo que permite la implementación del modelo a instancias más pequeñas para cada cluster por separado. Para realizar la división en clusters, es necesario identificar características en común entre las líneas de envasado y los productos que éstas procesan, las que pueden ser debido al tipo de jarabe o formato. Cada línea, tanque y producto debe estar en sólo un cluster, de tal manera que el total de la planta es dividida y el modelo es aplicado en cada cluster. Si bien esta división disminuye la cantidad de variables usadas en el modelo, cada cluster puede seguir siendo difícil de resolver para grandes plantas que producen una gran variedad de productos, como en este caso. Es por esto que se aplica la segunda fase de esta metodología.

3.3.2. Programación de productos a priori

En esta segunda fase, antes de aplicar el modelo, se realiza una programación a priori de un subconjunto de los productos a programar asignados en cada cluster. Esta programación a priori se realiza fijando las variables que determinan la secuencia de estos productos en las líneas que pertenecen a un cluster, es decir, fijando en 1 algunas de las variables x_{ijm}^L , ya que es en las líneas donde se define el tiempo total de proceso de cada producto debido a que un producto sólo puede ser producido en una sola línea. El criterio utilizado para realizar esta programación a priori corresponde a una programación en orden ascendente del tiempo de entrega de los productos, debido a las prioridades de la firma de cumplir con los deadlines de los productos. Este criterio es implementado de manera que minimice el makespan, es decir, se programan los productos en la línea que se desocupa primero,

siempre y cuando esa línea pueda procesar ese producto. Para hacer esto, primero se deben ordenar todos los productos a programar por orden ascendente de su tiempo de entrega, luego se van asignando los productos a las líneas en ese orden. Para asignar un producto a alguna de las líneas, se debe calcular el makespan en cada caso por separado, es decir, se asigna primero a una línea y se calcula el makespan, luego se asigna a otra línea y se calcula nuevamente el makespan y así sucesivamente; luego, el producto queda asignado a la línea en que se produjo el menor makespan. Para determinar la cantidad de productos a programar a priori, se resolvió el modelo con todos los productos programados de esta manera e iterativamente se disminuyó la cantidad productos programados a priori desde su última posición, resolviendo el modelo hasta que se obtuvo una solución factible dentro del tiempo límite y un GAP aceptable por el programador.

Con la estrategia de resolución propuesta es posible programar el total de producción y explotar el modelo para la programación de los productos en los tanques, ya que es en esta etapa donde existen los mayores tiempos de set up.

Para realizar una evaluación del criterio utilizado para la programación a priori, se aplicó un segundo criterio para realizar una comparación entre ambos. Este nuevo criterio corresponde a realizar la programación a priori por orden descendente del tiempos de proceso de los productos, aplicado de la misma manera que el primer criterio. Este nuevo criterio es muy utilizado en la programación de actividades y ayuda a la disminución del makespan, debido a que equilibra la carga entre las máquinas para disminuir el tiempo total de proceso. Sin embargo, es posible que al utilizar este nuevo criterio algunos de los productos con tiempo de entrega impuesto, no puedan cumplirlo, debido a que este nuevo criterio sólo ordena según su tiempo de proceso. Es por esto que al aplicar este criterio se debe relajar la restricción de deadline, por lo que se deben realizar pequeñas modificaciones al modelo propuesto. Primero, se incluye una nueva variable de holgura al modelo, que representará el tiempo de retraso en la completación del producto (tardiness), por lo que se define una nueva variable R_j y la restricción (3.23) debe ser reemplazada por la siguiente:

R_j Retraso en el tiempo de completación del producto j .

$$d_j + R_j \geq C_j^L \quad j \in J_0 \quad (3.32)$$

En esta nueva restricción, la nueva variable añadida al modelo tiene como objetivo relajar los tiempos de entrega de los productos y determinar el retraso del mismo. Además de esto, como no es deseable que el retraso sea demasiado, se agrega esta variable a la función objetivo para que sea minimizada, por lo que la función objetivo del modelo propuesto (3.1) debe ser reemplazada por la siguiente:

$$\text{Min} \quad C_{\text{máx}} + \sum_j R_j \quad (3.33)$$

Esta nueva función objetivo incluye la minimización del makespan junto con la suma de los posibles retrasos de cada producto, ambos medidos en tiempo. De esta manera, se mantiene la estructura inicial y se incorpora la relajación de los tiempos de entrega penalizada en la función objetivo, con lo que se permiten los retrasos pero asociados a un mayor valor objetivo. En resumen, la estrategia de aplicación del modelo sigue el siguiente procedimiento:

Estrategia de resolución basado en la reducción del problema y modelo MIP

- Paso 1: Crear clusters, asignando máquinas y productos a cada cluster, de manera de abordar el total de la planta.
- Paso 2: Programar a priori un subconjunto de los productos según criterio mediante la fijación de variables en cada cluster.
- Paso 3: Resolver el modelo (3.1) - (3.31) (o modelo modificado, según corresponda) en cada cluster.

4 | Resultados

4.1. Aplicación al caso de estudio

En esta sección se describe la aplicación del modelo y su metodología a un problema real de una compañía embotelladora de bebidas no alcohólicas en Chile. El desarrollo, aplicación y resultados del presente trabajo fueron realizados en esta empresa durante 6 meses, donde también se recopilaron los datos evaluados. Se trabajó específicamente con el sub gerente de Planificación y Logística de la planta, quien junto con sus trabajadores proporcionaron toda la información necesaria acerca del proceso productivo, planes de producción, datos del proceso, softwares y métodos utilizados. También se visitaron las salas de elaboración y las líneas de producción, donde se consultó también con los operadores de las máquinas. Toda la información recabada se utilizó para la construcción y validación del modelo propuesto y su estrategia de resolución.

Para la aplicación de la metodología descrita, la primera fase corresponde a la división de la planta en clusters. Para hacer esto, los productos con ciertas características en común fueron agrupados. El total de productos que la compañía produce puede ser clasificado en dos grandes grupos: Bebidas carbonatadas y Bebidas no carbonatadas. Las bebidas carbonatadas son aquellas que pasan por un proceso de carbonatación que produce el efecto burbujeante, el cual implica la introducción de varios ingredientes, entre ellos dióxido de carbono (CO_2). En el caso de las bebidas no carbonatadas, se incluyen los jugos néctar y bebidas funcionales como energéticas y bebidas isotónicas. Los tiempos de set up entre estos dos tipos de bebidas son bastante altos, ya que requieren rigurosos procesos de limpieza

de las máquinas, por lo que la compañía tiene líneas de envasado dedicadas para estos dos tipos de bebidas. Además de esto, existen productos que sólo pueden ser producidos en líneas específicas, debido a su formato, como por ejemplo, productos envasados en latas.

Debido a las características de la firma mencionadas anteriormente, y siguiendo el Paso 1 de la estrategia, los clusters considerados son: Bebidas no carbonatadas (Cluster tipo 1), Bebidas carbonatadas (Cluster tipo 2) y Líneas específicas (Cluster tipo 3). Estos clusters definen de por sí una cantidad de productos asociados a cada uno, por lo que aún se deben asignar las máquinas a considerar en cada cluster. Para lograr esto, es necesario tener en cuenta la capacidad total de las máquinas consideradas y la cantidad de variables en el modelo que esto conlleva. Luego de experimentar con diferentes tamaños de modelo manteniendo una capacidad adecuada, los clusters tipo 1 y 2 tendrán asignados 3 tanques y 2 líneas cada uno. El cluster tipo 3 tendrá asignado 3 tanques y 1 línea. Para atender el total de la planta, se requieren dos clusters del tipo 1, tres clusters del tipo 2 y cinco clusters del tipo 3, pero los resultados aquí presentes sólo muestran la resolución de un cluster de cada tipo, ya que el objetivo es evaluar el desempeño del modelo propuesto. Dado lo anterior, los clusters considerados y sus tamaños se resumen en la Tabla 4.1.

Tabla 4.1: División de clusters para el caso de estudio.

	Cantidad	Descripción	Tanques	Líneas
Cluster tipo 1	2	Bebidas no carbonatadas	3	2
Cluster tipo 2	3	Bebidas carbonatadas	3	2
Cluster tipo 3	5	Línea específica	3	1

El modelo es resuelto para cada cluster implementando la segunda fase de la metodología (Sección 3.3.2) siguiendo el Paso 2 de la estrategia, luego los resultados son comparados con lo que actualmente hace la compañía. Esta metodología fue aplicada a tres instancias de una semana como horizonte de planificación, con datos reales de la compañía. En algunas semanas, hay líneas que no operan debido a que están en mantención, por lo que no necesariamente son las mismas líneas las que forman un cluster determinado en diferentes semanas. La Tabla 4.2 muestra los tamaños de cada instancia resuelta, junto con la cantidad

de productos a programar.

Tabla 4.2: Tamaño de las instancias de los clusters en cada semana.

Semana-Tipo de cluster	Cantidad de productos	Número de variables	Número de restricciones
1-1	21	2907	3910
1-2	15	1933	2926
1-3	11	1049	1245
2-1	9	881	1280
2-2	18	2584	3940
2-3	14	1496	1749
3-1	18	2260	2932
3-2	15	1733	2342
3-3	16	1834	2125

El modelo fue programado en lenguaje AMPL (ver anexo A) y resuelto en CPLEX 12.4.0.0 en un computador Intel Core i5 con un procesador de 2.6 GHz, siguiendo el Paso 3 de la estrategia. El criterio de parada es definido para soluciones con un error (GAP) menor o igual a 10 %, dentro del tiempo límite de 18000 segundos (5 horas). En la práctica, la compañía puede correr el programa mucho más tiempo, ya que la programación semanal es realizada con varios días de anticipación. Para reducir el tiempo de resolución del modelo, se utiliza la solución de la compañía como límite superior para la solución del modelo en cada instancia, es decir, el solver no divide los nodos del Branch-and-Bound que contienen una solución mayor que el de la compañía, por lo que cada solución arrojada por el modelo es mejor que el de la compañía. Esta solución de la compañía es usada como solución factible inicial para el modelo, a través de su valor en la función objetivo y se calcula de acuerdo a la programación propia de la empresa, la cual programa los productos según su indicador de Días Venta y prioridades de elaboración, esto es, cada producto tiene un orden de prioridad con respecto a las líneas de envasado que pueden producirlo. Este orden de prioridad tiene relación con la facilidad y velocidad con la que una línea de producción puede elaborar ese producto y determina en qué línea serán asignados los productos. Los resultados del modelo y su aplicación son resumidos en la Tabla 4.3.

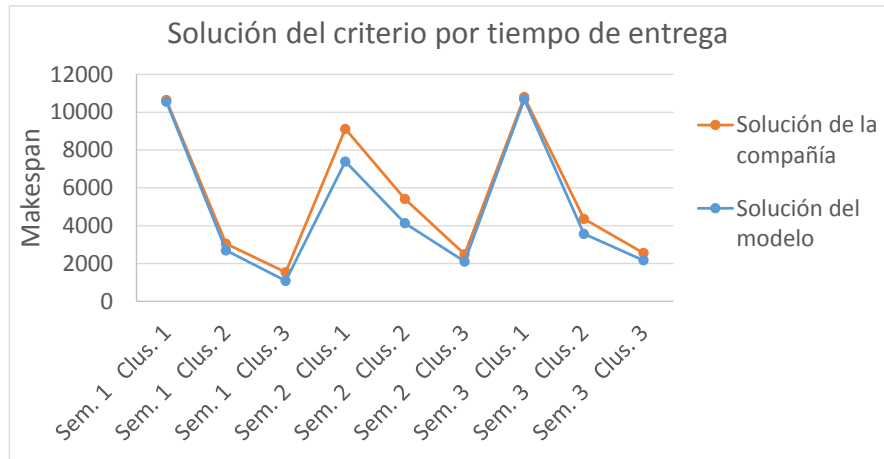
Tabla 4.3: Resultados computacionales para las instancias del caso de estudio.

Instancias	Criterio para la programación a priori	Productos a programar	Productos programados a priori	Productos con retraso	Makespan	GAP [%]	Tiempo [min]	Solución de la compañía	Mejora sobre la compañía [%]	
Semana 1	Cluster 1	Tiempo de entrega Tiempo de proceso	21	13 15	0 0	10550 9551	8.97 9.81	277 33.0	10640	0.85 10.24
	Cluster 2	Tiempo de entrega Tiempo de proceso	15	7 7	0 0	2695 2684	9.97 9.99	11.4 26.4	3044	11.47 11.83
	Cluster 3	Tiempo de entrega Tiempo de proceso	11	2 2	0 0	1086 1086	9.99 9.94	8.4 5.4	1535	29.25 29.25
Semana 2	Cluster 1	Tiempo de entrega Tiempo de proceso	9	4 4	0 1	7389 7175	9.62 11.83	216 300	9117	18.95 21.30
	Cluster 2	Tiempo de entrega Tiempo de proceso	18	14 15	0 0	4131 4534	9.84 9.50	9.0 3.0	5411	23.66 16.21
	Cluster 3	Tiempo de entrega Tiempo de proceso	14	7 6	0 0	2102 1988	9.98 9.94	4.8 7.2	2492	15.65 20.23
Semana 3	Cluster 1	Tiempo de entrega Tiempo de proceso	18	15 15	0 2	10678 8806	6.15 9.70	14.4 25.2	10798	1.11 18.45
	Cluster 2	Tiempo de entrega Tiempo de proceso	15	7 7	0 0	3570 3612	9.99 9.99	43.1 90.49	4361	18.14 17.18
	Cluster 3	Tiempo de entrega Tiempo de proceso	16	8 8	0 0	2167 2312	10.00 9.99	36.0 9.6	2555	15.19 9.51

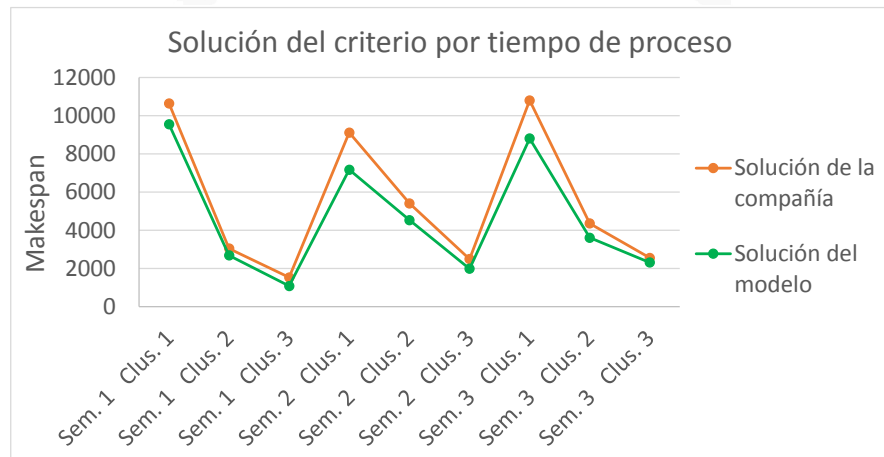
En esta tabla, la primera columna define las instancias resueltas y en la segunda columna se subdivide cada instancia por el cluster resuelto. En la tercera columna se describe el criterio utilizado en la programación a priori para resolver cada cluster. En la cuarta columna se muestra la cantidad de productos a programar en cada cluster y en la quinta columna se muestra la cantidad de productos programados a priori utilizando el criterio señalado. La sexta columna muestra la cantidad de productos que fueron terminados después de su tiempo de entrega. Cabe destacar que sólo existe retraso cuando se utiliza el criterio por tiempo de proceso, ya que en este caso se utiliza el modelo que relaja los deadlines. En las columnas 7, 8 y 9 se muestran el valor del Makespan alcanzado con la metodología propuesta, su GAP asociado y el tiempo de resolución, respectivamente. Para realizar la comparación de los resultados con lo que actualmente hace la compañía, en las dos últimas columnas se muestran la solución de la compañía y el porcentaje de mejora, respectivamente, que se logra con la metodología propuesta.

4.2. Análisis de resultados

Primero que todo, es importante mencionar que los clusters definidos tienen ciertas características que afectan el desempeño del modelo en cada uno de ellos, por lo que un cluster puede ser más difícil de resolver que otro. Las bebidas no carbonatadas requieren un intensivo uso de los tanques de acumulación, debido a que todo el contenido de una botella de estos productos son preparados en los tanques, a diferencia de las bebidas carbonatadas, donde la mayor parte de una botella corresponde a agua, la que es añadida a la botella directamente en las líneas. Esto hace que los productos del Cluster 1 usen en gran cantidad la división de lotes en los tanques comparado con los Clusters 2 y 3, por lo que es esperable que el Cluster 1 requiera mayor tiempo de resolución y necesite programar más productos a priori. A pesar de esto, los resultados muestran que las soluciones son capaces de superar a la compañía en todas las instancias, lo que muestra la factibilidad de usar modelos complejos junto con una apropiada metodología de implementación, como la presentada en este trabajo.

**Figura 4.1:** Soluciones del criterio por tiempo de entrega.

(Fuente: Elaboración propia.)

**Figura 4.2:** Soluciones del criterio por tiempo de proceso.

(Fuente: Elaboración propia.)

Si bien los dos criterios utilizados para la programación a priori superaron la solución de la compañía en todas las instancias (ver Figura 4.1 y 4.2), al ser comparados entre ellos, los resultados muestran que el criterio por tiempos de proceso fue mejor en 5 de 9 oportunidades, obteniendo una mejora promedio sobre la compañía de 16,72 %, mientras que el criterio por tiempo de entrega obtuvo un promedio de 14,61 %, por lo que ambos tienen un comportamiento similar. Si bien existen dos criterios diferentes, ambos son aplicados de manera de minimizar el Makespan, como se mencionó antes en la sección 3.3, lo que explicaría sus resultados similares. Cabe destacar que los valores de la solución

de la compañía están sobrestimados, ya que, si bien se programan los productos en las líneas según las reglas de la empresa, ésta no tiene reglas al momento de programar los tanques, por lo que la solución de la compañía es entregada por el modelo, fijando sólo la programación en las líneas, utilizando una programación óptima en los tanques.

Con respecto a la cantidad de productos programados a priori, ésta varía en relación al tamaño de la instancia, donde en promedio se debió programar a priori un 53,66 % del total de productos a programar (ver Figura 4.3), lo que prueba la dificultad de resolver el modelo para grandes instancias debido a su complejidad.

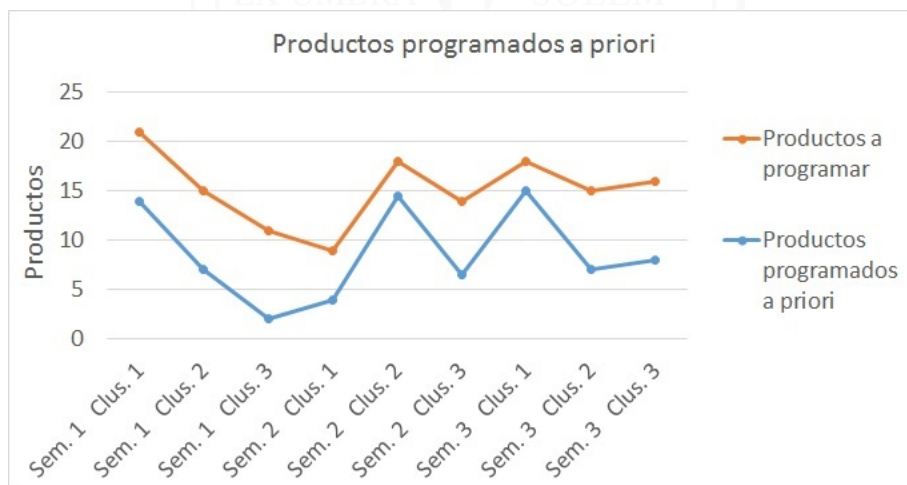


Figura 4.3: Cantidad de productos programados a priori en cada instancia.

(Fuente: Elaboración propia.)

Sin embargo, no es fácil determinar la cantidad de productos a programar a priori, debido a la sensibilidad de los tiempos de resolución con respecto a esto. En la Figura 4.4 se puede observar que a medida que se disminuye la cantidad de productos fijados a priori, el tiempo de resolución crece muy rápidamente, superando fácilmente el tiempo límite de 5 horas. Si bien se definió una manera de determinar la cantidad de productos a fijar (sección 3.3), hay casos en que se debe resolver muchas veces el modelo, lo que toma una gran cantidad de recursos en memoria RAM, por lo que también el programador puede decidir sacrificar optimalidad aumentando los productos a fijar para obtener una solución en menor tiempo.

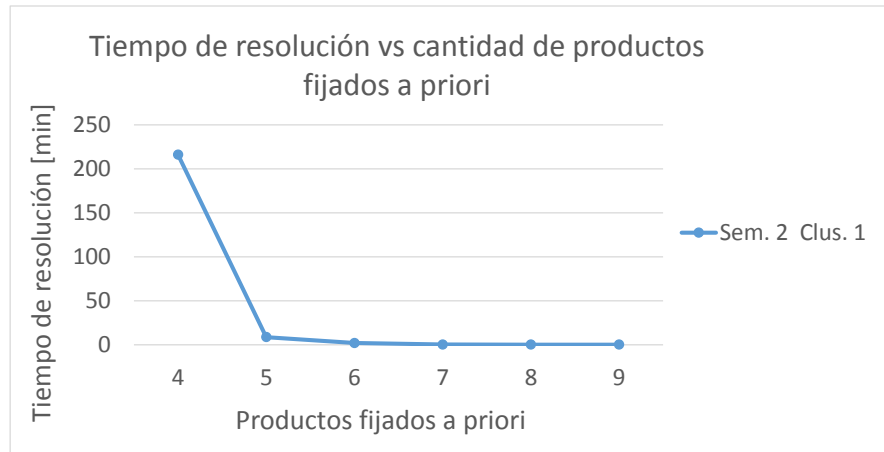


Figura 4.4: Sensibilidad del modelo en el tiempo de resolución con respecto a la cantidad de productos fijados a priori.

(Fuente: Elaboración propia.)

Al igual que existe gran sensibilidad del tiempo con respecto a la cantidad de productos programados a priori, también existe sensibilidad del tiempo con respecto al GAP utilizado como límite en el modelo. En la presentación de los resultados se utilizó un GAP del 10 % ya que, además de ser un valor aceptable por la compañía, utilizar menores valores significaba un excesivo aumento del tiempo de resolución, como se puede observar en la Figura 4.5.

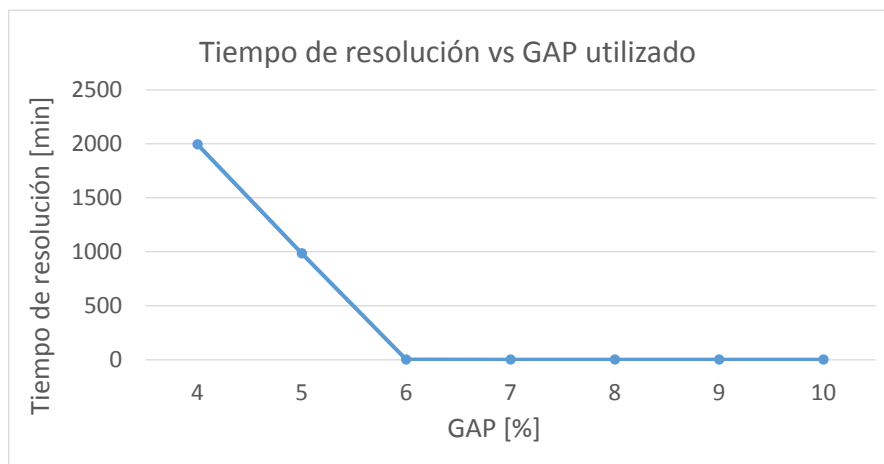
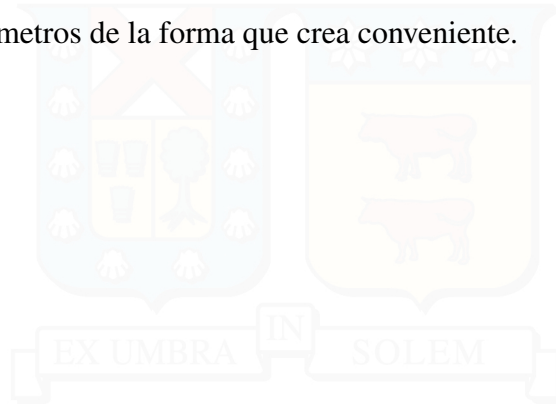


Figura 4.5: Sensibilidad del modelo en el tiempo de resolución con respecto al GAP utilizado.

(Fuente: Elaboración propia.)

Los parámetros asociados a la aplicación del modelo y sus heurísticas como los criterios de parada, creación de clusters o cantidad de productos a fijar a priori dependen de las características de cada compañía y también de las características del computador utilizado para la resolución del modelo, por lo que el programador tiene ciertos grados de libertad para definir estos parámetros de la forma que crea conveniente.



5 | Conclusiones

En este trabajo se abordó la programación de la producción a corto plazo para una industria de bebidas no alcohólicas. Se presentó un modelo de programación lineal entera mixta y una estrategia de resolución ad-hoc para un problema real de una compañía chilena. El modelo considera la minimización del tiempo total de producción de bebidas no alcohólicas en dos etapas sincronizadas y está basado en el Problema del Vendedor Viajero (TSP).

La estrategia de resolución contempla dos fases: en la primera se divide la planta en clusters, y en la segunda se realiza una programación a priori de una parte de los productos a programar utilizando dos criterios diferentes. Estos dos criterios obtuvieron similares resultados, por lo que la aplicación de cada uno dependerá de las prioridades de cada compañía. A pesar del hecho de que en promedio se debió programar casi la mitad de los productos a priori (53,66 %), se obtuvieron mejores soluciones que los que obtuvo la compañía en esas semanas, logrando mejoras de hasta un 30 % en los mejores casos.

La metodología propuesta permite la utilización de sofisticados modelos de optimización para resolver problemas reales de una compañía. Los resultados muestran que en promedio se logró mejorar en 15,67 % la solución de la compañía, lo que prueba los beneficios de usar herramientas de optimización como esta.

5.1. Investigaciones futuras

Si bien el este trabajo presenta resultados superiores a los de la compañía, también muestra dificultades y complejidad en su resolución, por lo que se pueden encontrar mejores maneras para plantear y resolver el modelo propuesto.

En futuras investigaciones se pueden buscar nuevas reformulaciones del modelo propuesto, utilizando nuevas técnicas y planteamientos, como lo es la utilización de un modelo en tiempo discreto, a diferencia del propuesto en este trabajo, que es de tiempo continuo. En este tipo de modelos, el tiempo es subdividido en pequeños micro-períodos, lo que permite plantear un modelo en que la programación se realiza sólo con variables enteras, dado que simplemente se debe indicar si un micro-período es utilizado o no. Pueden existir diferentes maneras en la construcción de los micro-períodos, lo que también afecta la complejidad del modelo.

Además de lo anterior, pueden utilizarse otros métodos para la resolución del modelo en vez de las heurísticas de reducción del problema presentados, como lo son la búsqueda de inecuaciones válidas que permitan disminuir el espacio de soluciones factibles.

Un método muy utilizado para resolver problemas del tipo NP-hard como éste, son las llamadas metaheurísticas, las cuales en vez de utilizar un modelo de programación matemática para encontrar la solución, utilizan un método de generación de soluciones factibles que son evaluadas y luego mejoradas para llegar a una solución lo suficientemente buena. En los problemas de programación es muy común la utilización de metaheurísticas como algoritmos genéticos, greedy randomized adaptive search procedure (GRASP), simulated annealing, búsqueda tabú, aceptación por umbrales, entre otras.

También es posible plantear la resolución de un problema utilizando una combinación de estas metaheurísticas con la programación matemática, de manera que exista intercambio de información entre ellas. Una opción es dividir el conjunto de las variables de un modelo

para que algunas sean resultado de una metaheurística y luego se resuelve un modelo reducido para determinar el otro conjunto de variables. De esta manera se utilizan las ventajas de ambos métodos para la resolución del problema.



Bibliografía

- Allahverdi, Ali; Ng, CT; Cheng, TC Edwin; y Kovalyov, Mikhail Y (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 187(3), 985–1032. [2.2](#)
- Chen, TsiuShuang; Long, Lei; y Fung, Richard YK (2006). A genetic algorithm for the batch scheduling with sequence-dependent setup times. In *Intelligent Computing in Signal Processing and Pattern Recognition* (pp. 1137–1144). Springer. [2.2](#)
- Copil, Karina; Wörbelauer, Martin; Meyr, Herbert; y Tempelmeier, Horst (2016). Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR Spectrum*, (pp. 1–64). [2.2](#)
- Ferreira, Deisemara; Clark, Alistair R; Almada-Lobo, Bernardo; y Morabito, Reinaldo (2012). Single-stage formulations for synchronised two-stage lot sizing and scheduling in soft drink production. *International Journal of Production Economics*, 136(2), 255–265. [2.2](#)
- Ferreira, Deisemara; Morabito, Reinaldo; y Rangel, Socorro (2009). Solution approaches for the soft drink integrated production lot sizing and scheduling problem. *European Journal of Operational Research*, 196(2), 697–706. [2.2](#), [3.2](#)
- Fleischmann, Bernhard y Meyr, Herbert (1997). The general lotsizing and scheduling problem. *Operations-Research-Spektrum*, 19(1), 11–21. [2.2](#)
- Graham, Ronald L; Lawler, Eugene L; Lenstra, Jan Karel; y Kan, AHG Rinnooy (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5, 287–326. [2.1.1](#)
- Hnaïen, Faïcel; Yalaoui, Farouk; y Mhadhbi, Ahmed (2015). Makespan minimization on a two-machine flowshop with an availability constraint on the first machine. *International Journal of Production Economics*, 164, 95–104. [2.2](#)
- Jia, Zhao-hong; Li, Kai; y Leung, Joseph Y-T (2015). Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities. *International Journal of Production Economics*, 169, 1–10. [2.2](#)
- Karmakar, Sandipan y Mahanty, Biswajit (2010). Minimizing makespan for a flexible flow shop scheduling problem in a paint company. *Industrial Engineering and Operations Management*. [2.2](#)

- Kurz, Mary E y Askin, Ronald G (2004). Scheduling flexible flow lines with sequence-dependent setup times. *European Journal of Operational Research*, 159(1), 66–82. [2.2](#)
- Leiva, Javier Cuzmar y Albornoz, Víctor M. (2016). Short-term production scheduling in the soft drink industry. In *Proceedings of 5th the International Conference on Operations Research and Enterprise Systems* (pp. 416–423). [3](#)
- Meyr, Herbert (2000). Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, 120(2), 311–326. [2.2](#)
- Meyr, Herbert (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, 139(2), 277–292. [2.2](#)
- Pinedo, Michael (2015). *Scheduling*. Springer. [2.1](#), [2.2](#)
- Radhakrishnan, Sanjay y Ventura, Jose A (2000). Simulated annealing for parallel machine scheduling with earliness-tardiness penalties and sequence-dependent set-up times. *International Journal of Production Research*, 38(10), 2233–2252. [2.1.3](#)
- Ríos-Mercado, Roger Z y Bard, Jonathan F (1998). Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Computers & Operations Research*, 25(5), 351–366. [2.2](#)
- Ríos-Mercado, Roger Z y Bard, Jonathan F (2003). The flow shop scheduling polyhedron with setup times. *Journal of Combinatorial Optimization*, 7(3), 291–318. [2.1.3](#)
- Rocha, Pedro Leite; Ravetti, Martín Gómez; Mateus, Geraldo Robson; y Pardalos, Panos M (2008). Exact algorithms for a scheduling problem with unrelated parallel machines and sequence and machine-dependent setup times. *Computers & Operations Research*, 35(4), 1250–1264. [2.2](#)
- Ruiz, Rubén y Andrés-Romano, Carlos (2011). Scheduling unrelated parallel machines with resource-assignable sequence-dependent setup times. *The International Journal of Advanced Manufacturing Technology*, 57(5-8), 777–794. [2.2](#)
- Seeanner, Florian y Meyr, Herbert (2013). Multi-stage simultaneous lot-sizing and scheduling for flow line production. *OR spectrum*, 35(1), 33–73. [2.2](#)
- Toledo, Claudio Fabiano Motta; Arantes, Márcio da Silva; y França, Paulo Morelato (2011). Tabu search to solve the synchronized and integrated two-level lot sizing and scheduling problem. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (pp. 443–448).: ACM. [2.2](#)
- Toledo, Claudio Fabiano Motta; de Oliveira, Lucas; de Freitas Pereira, Rodrigo; França, Paulo Morelato; y Morabito, Reinaldo (2014). A genetic algorithm/mathematical programming approach to solve a two-level soft drink production problem. *Computers & Operations Research*, 48, 40–52. [2.2](#)

Yilmaz Eroglu, Duygu y Ozmutlu, H Cenk (2014). Mip models and hybrid algorithms for simultaneous job splitting and scheduling on unrelated parallel machines. *The Scientific World Journal*, 2014. [2.2](#)



A | Código AMPL (.mod)

CONJUNTOS

```

set TOTALSKU;                # Total de productos.
set maquinas;                # Total de máquinas (tanques y líneas).
set LOTES within {TOTALSKU}; # Productos a programar.
set LOTES0 := LOTES union {0}; # Conjunto de productos a programar
                                # incluido el producto ficticio.

set tanques within {maquinas}; # Tanques a programar.
set lineas within {maquinas};  # Líneas a programar.
set alfa within {maquinas,TOTALSKU}; # Conjunto de pares entre máquinas que
                                # pueden producir los productos.

set beta := setof {m in tanques, j in LOTES0: (m,j) in alfa} (m,j);
                                # Conjunto de pares entre tanques a
                                # programar que pueden producir los
                                # productos a programar.

set gamma := setof {m in lineas, j in LOTES0: (m,j) in alfa} (m,j);
                                # Conjunto de pares entre líneas a
                                # programar que pueden producir los
                                # productos a programar.

```

PARÁMETROS

```

param D {LOTES} >= 0;        # Demanda.
param a {LOTES,lineas} >= 0; # Tiempo de proceso por unidad.
param sT {LOTES0,LOTES0} >= 0; # Tiempos de set up en tanques.
param sL {LOTES0,LOTES0} >= 0; # Tiempos de set up en líneas.
param d {LOTES0} >= 0;      # Deadline.

```

```

param r {LOTES} >= 0;          # Cantidad de litros por unidad.
param K {tanques} >=0;        # Capacidad de tanques.
param O {j in LOTES0} := card(setof {m in tanques: (m,j) in beta} (m,j));
                                # Cardinalidad del conjunto beta.
param G := sum {j in LOTES} ((max{i in LOTES0} sT[i,j]) + (max {w in lineas:(w,j)
    in gamma} a[j,w])*D[j] + floor(r[j]*D[j]/(min{m in tanques: (m,j) in beta}
    K[m]))*sT[j,j]);          # Límite superior del tiempo de
                                # completación de los productos.

### VARIABLES ###

var CT {j in LOTES0, m in tanques, o in 1..O[j]: (m,j) in beta} >=0;
                                # Tiempo de completación tanques.
var CL {LOTES0} >=0;          # Tiempo de completación líneas.
var yT {j in LOTES0, m in tanques, o in 1..O[j]: (m,j) in beta} binary;
                                # Asignación productos en tanques.
var yL {j in LOTES0, m in lineas: (m,j) in gamma} binary;
                                # Asignación productos en líneas.
var xT {i in LOTES0,j in LOTES0,m in tanques: (m,i) in beta and (m,j) in beta}
    binary;                    # Secuenciamiento en tanques.
var xL {i in LOTES0, j in LOTES0, m in lineas: (m,i) in gamma and (m,j) in gamma}
    binary;                    # Secuenciamiento en líneas.
var pT {j in LOTES, m in tanques, o in 1..O[j]: (m,j) in beta} >=0;
                                # Tiempo de proceso en tanques.
var v {j in LOTES} >=0;        # Tiempo de espera en líneas.
var q {j in LOTES, m in tanques, o in 1..O[j]: (m,j) in beta} integer >=0;
                                # Número de rellenos en tanques.
var Cmax >=0;                  # Makespan

```

```
### FUNCIÓN OBJETIVO ###
```

```
minimize Makespan:
```

```
    Cmax;
```

RESTRICCIONES EN TANQUES

subject to tanques1 {j in LOTES}:

sum {m in tanques, o in 1..0[j]: (m,j) in beta} yT[j,m,o] >= 1;

subject to tanques2 {m in tanques}:

sum {o in 1..0[0]} yT[0,m,o] = 1;

subject to tanques3 {i in LOTES0, m in tanques: (m,i) in beta}:

sum {j in LOTES0: (m,j) in beta} xT[i,j,m] = sum {o in 1..0[i]} yT[i,m,o];

subject to tanques4 {j in LOTES0, m in tanques: (m,j) in beta}:

sum {i in LOTES0: (m,i) in beta} xT[i,j,m] = sum {o in 1..0[j]} yT[j,m,o];

subject to tanques5 {i in LOTES0, j in LOTES, w in lineas, m in tanques: (m,i) in beta and (m,j) in beta and (w,j) in gamma}:

sum {o in 1..0[j]} CT[j,m,o] + G*(2 - xT[i,j,m] - yL[j,w]) >= sum {o in 1..0[i]} CT[i,m,o] + sum {o in 1..0[j]} sT[i,j]*yT[j,m,o] + sum {o in 1..0[j]} a[j,w]*pT[j,m,o]/r[j] + sum {o in 1..0[j]} q[j,m,o]*sT[j,j];

subject to tanques6 {j in LOTES}:

sum {m in tanques, o in 1..0[j]: (m,j) in beta} pT[j,m,o] >= r[j]*D[j];

subject to tanques7 {j in LOTES, m in tanques, o in 1..0[j]: (m,j) in beta}:

pT[j,m,o] <= r[j]*D[j]*yT[j,m,o];

subject to tanques8 {j in LOTES, m in tanques, o in 1..0[j]: (m,j) in beta}:

CT[j,m,o] <= G*yT[j,m,o];

subject to tanques9 {j in LOTES, m in tanques: (m,j) in beta}:

sum {o in 1..0[j]} yT[j,m,o] <= 1;

subject to tanques10 {j in LOTES, o in 1..0[j]}:

sum {m in tanques: (m,j) in beta} yT[j,m,o] <= 1;

```

subject to tanques11 {j in LOTES, o in 2..O[j]}:
    sum {m in tanques: (m,j) in beta} yT[j,m,o-1] >= sum {m in tanques: (m,j)
    in beta} yT[j,m,o];

subject to tanques12 {j in LOTES, m in tanques, o in 1..O[j]: (m,j) in beta}:
    q[j,m,o] <= pT[j,m,o]/(K[m]);

subject to tanques13 {j in LOTES, m in tanques, o in 1..O[j]: (m,j) in beta}:
    q[j,m,o] >= pT[j,m,o]/(K[m]) - 1;

subject to tanques14 {i in LOTES, m in tanques: (m,i) in beta}:
    xT[i,i,m] = 0;

### RESTRICCIONES EN LÍNEAS ###

subject to lineas1 {j in LOTES}:
    sum {m in lineas: (m,j) in gamma} yL[j,m] = 1;

subject to lineas2 {m in lineas}:
    yL[0,m] = 1;

subject to lineas3 {i in LOTES0, m in lineas: (m,i) in gamma}:
    sum {j in LOTES0: (m,j) in gamma} xL[i,j,m] = yL[i,m];

subject to lineas4 {j in LOTES0, m in lineas: (m,j) in gamma}:
    sum {i in LOTES0: (m,i) in gamma} xL[i,j,m] = yL[j,m];

subject to lineas5 {i in LOTES0, j in LOTES, m in lineas: (m,i) in gamma and (m,j)
    in gamma}:
    CL[j] + G*(1 - xL[i,j,m]) >= CL[i] + sL[i,j] + a[j,m]*D[j] + v[j];

subject to lineas6 {j in LOTES0}:
    Cmax >= CL[j];

subject to lineas7 {i in LOTES0}:
    d[i] >= CL[i];

```

```

subject to lineas8 {i in LOTES, m in lineas: (m,i) in gamma}:
    xL[i,i,m] = 0;

```

RESTRICCIONES DE SINCRONIZACION

```

subject to sincronizacion1 {j in LOTES, w in lineas: (w,j) in gamma}:
    CL[j] - a[j,w]*D[j] - v[j] <= G*(1 - yL[j,w]) + sum {m in tanques: (m,j)
    in beta} CT[j,m,1] - sum {m in tanques: (m,j) in beta} a[j,w]*pT[j,m,1]/
    r[j] - sum {m in tanques: (m,j) in beta} q[j,m,1]*sT[j,j];

```

```

subject to sincronizacion2 {j in LOTES, o in 2..O[j], w in lineas: (w,j) in
gamma}:
    sum {m in tanques: (m,j) in beta} CT[j,m,o] + G*(2 - sum {m in tanques:
(m,j) in beta} yT[j,m,o] - yL[j,w]) >= sum {m in tanques: (m,j) in beta}
CT[j,m,o-1] + sum {m in tanques: (m,j) in beta} a[j,w]*pT[j,m,o]/r[j] +
sum {m in tanques: (m,j) in beta} q[j,m,o]*sT[j,j];

```

```

subject to sincronizacion3 {j in LOTES, m in tanques: (m,j) in beta}:
    CL[j] >= sum {o in 1..O[j]} CT[j,m,o];

```