ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

ESTIMACIÓN DE TIEMPOS DE ENTREGA DE PEDIDOS EN UNA EMPRESA DE LA INDUSTRIA MANUFACTURERA USANDO PROGRAMACIÓN MATEMÁTICA

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIEROS MATEMÁTICOS

PROYECTO DE INVESTIGACIÓN

LUIS SEBASTIÁN ALCÍVAR RODRÍGUEZ
sebasalcivar08@gmail.com
Y
RAFAEL ALEJANDRO CHÁVEZ RIERA
rchavezriera@gmail.com

Director: DIEGO RECALDE CALAHORRANO, PH.D.
diego.recalde@epn.edu.ec

QUITO, JULIO 2019

DECLARACIÓN

Nosotros LUIS SEBASTIÁN ALCÍVAR RODRÍGUEZ y RAFAEL ALEJANDRO CHÁVEZ RIERA, declaramos bajo juramento que el trabajo aquí escrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos de propiedad intelectual, correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

Luis Sebastián Alcívar Rodríguez Rafael Alejandro Chávez Riera

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por LUIS SEBASTIÁN ALCÍVAR RODRÍGUEZ y RAFAEL ALEJANDRO CHÁVEZ RIERA, bajo mi supervisión.

Diego Recalde Calahorrano, Ph.D.
Director del Proyecto

AGRADECIMIENTOS

De Sebastián:

Este trabajo está dedicado especialmente a mi familia; Gus, Carmita, Ale y Alfon por todo el apoyo y cariño a lo largo de mi vida.

A mis amigos, compañeros, colegas, maestros y conocidos de la universidad que han estado para estudiar y también para no estudiar, pero han estado. A aquellos que siempre estaban para un partido de volley o fútbol. A todas las personas que tuve el agrado de conocer a lo largo de mi carrera.

A todos los miembros del Club de Alto Rendimiento Chacaritas, algún día jugaremos nuevamente.

A mis compañeros de trabajo que me han ayudado a crecer como persona y como profesional.

A Diego Recalde por su ayuda en la elaboración de este trabajo.

A Paty por su amistad a lo largo de la carrera; y ahora, por cuestiones de la vida, una gran compañera de viaje.

A Rafa Chávez por compartir la experiencia de realizar una tesis.

AGRADECIMIENTOS

De Rafael

A Dios.

A mis ídolos: Alejandro Magno, Francisco José, Rafael Antonio, Mateo, Clara.

A Miguel Ángel y María del Carmen.

A José Miguel, Gabriel, Francisco e Inmaculada.

A Patricia, Carlos, Cecilia, Pepe, Solange, Marco, Mildres, Gaby, Francesco.

A Carlos, Julián, Daniel, Alberto, Carla, Lucía, Belén, Carolina, Andrea.

A Joffre, Feto, Nemo, Checo, Perra, Fernando, Marcelo, Tito, Yunke, Marmota, Negro, Bebo, Buddy y todos los demás.

A Ernesto, Tocayo, Cristian, Jimmy, Andrés, Alexander.

A los fans de los chacaritas.

A Ronny, Eduardo, Elías, Carlos, Paúl, Juan José, Pablo, Miguel Ángel, Geremy, Omar, Paola, Valeria, Gaby y los demás.

A Renato, Antonio, Arturo, Emilio, Ramiro, José Xavier, Nicolás, Esteban, Simón, María José, Sebastián, Martín.

A Eduardo, Virgilio, José Xavier, Lourdes.

A Lucas, Juan José, Juan Miguel, Antonio, José Andrés, Jimmy, Carlitos, Manuel Eduardo, Pancho, Pirata, Enrique, Fernando, Gianmarco.

Al país por la educación gratuita.

A los lectores por el tiempo que tuvieron que dedicarnos particularmente.

A Diego Recalde pues gracias especialmente a él logramos obtener el título.

A mis amigos de Vsoft que nos mantuvieron ejercitados.

A Christian, Enrique, Andre, Gijar, Juan Javier y Ana María.

A Fiódor, Oscar, John Ronald, Alexandre, Adam.

A Dorian, Bruce, Tony, Peter, Aragorn, Aliosha.

A Sebastián Alcívar, sin él no habría podido hacer este trabajo. Literalmente.

Las matemáticas pueden ser definidas como aquel tema del cual no sabemos nunca lo que decimos ni si lo que decimos es verdadero.

Bertrand Russell

Índice general

Resumen x

Abstract xi

1. Introducción 1

1.1. Motivación 2

1.2. Objetivos 3

1.3. Estado del arte 4

1.4. Programación dinámica del proceso de trabajo 5

1.5. Estructura del trabajo 5

2. Job Shop Scheduling Problem 7

2.1. Introducción 7

2.2. Terminología 7

2.3. Modelo 8

3. Estimación de tiempos de entrega 12

3.1. Introducción 12

3.2. Modelo 13

3.3. Heurísticas 15

4. Aplicación del modelo 19

4.1. Sobre los datos 19

4.2. Simulaciones 22

4.2.1. Instancia 10 25

4.2.2. Instancia 14 30
4.2.3. Instancia 19 37
4.3. Resultados 49

5. Conclusiones y Recomendaciones **65**
5.1. Conclusiones 65
5.2. Recomendaciones 66

A. Apéndice **68**
A.1. Tablas de tiempos 68
A.2. Códigos 71
A.2.1. Modelo Lineal 71
A.2.2. Heurística Base 74
A.2.3. Heurística OA 78
A.2.4. Modificación de Heurística de Ordenamiento Aleatorio para tiempos iniciales 82

Bibliografía **89**

Resumen

En el presente trabajo se implementa un modelo lineal y heurísticas para estimar tiempos de entrega de pedidos en industrias manufactureras. Se realiza una comparación entre el modelo lineal y las heurísticas para identificar la factibilidad de la aplicación de cada una de ellas a la realidad de la empresa manufacturera. Para ello se considera los tiempos de ejecución de cada uno y la aproximación que tienen las heurísticas a la solución óptima como factores de decisión. A través de simulaciones y datos reales de una empresa, se verifica dicha factibilidad.

Abstract

In this document a linear model and heuristics are implemented in order to estimate delivery times in manufacturing industries. A comparison between both the heuristics and the linear model is made to identify the feasibility of applying them to the reality of a company's organization. The heuristic's execution times and the approximation to the optimal solution are taken as decision factors for the aforementioned feasibility. This feasibility is checked through simulations and the business' real data applications of the model's and heuristics' algorithms.

Capítulo 1

Introducción

En la industria manufacturera uno de los problemas más difíciles de determinar es el tiempo que le toma a una empresa terminar un encargo de un cliente. Aunque se suele trabajar con una estimación empírica para acordar un tiempo de entrega con el cliente, frente a la competencia conviene reducir ese tiempo o, mejor, saber exactamente cuánto se demora la empresa en completar un encargo considerando la carga de trabajo actual. En muchas ocasiones, el tiempo será negociado y establecido según la urgencia del cliente, en cuyo caso, la industria tendrá que extender su jornada de trabajo para cumplir con las expectativas del cliente. En otro caso, simplemente tendrá que dejar a un lado los trabajos pendientes que sabe que terminará pronto o dejar a un lado aquellos trabajos de un cliente más flexible con los tiempos de entrega para culminar con los trabajos más urgentes.
En este trabajo, se estimarán los tiempos de entrega de pedidos de clientes en una empresa de manufactura considerando los trabajos pendientes en la misma. Para ello se tiene:

- m máquinas.
- n trabajos independientes, donde cada trabajo j debe ser procesado por solamente una de las m máquinas.
- Ninguna máquina realizará más de un trabajo al mismo tiempo.
- El tiempo de procesamiento de un trabajo es un número finito positivo fijo conocido.
- Cada máquina i ($i = 1, \ldots, m$) procesa un trabajo j en un tiempo $p_{ij} \geq 0$.

Considerando los parámetros definidos, denotemos a este problema como **P1**. El problema consiste en hallar el menor tiempo que les toma a las m máquinas terminar los n trabajos, considerando que, en cualquier instante t, puede ingresar un nuevo trabajo lo que implica reestructurar el orden de procesamiento para minimizar el tiempo de finalización del nuevo conjunto de trabajos.

1.1. Motivación

En la industria manufacturera, y en la industria en general, el problema de controlar los tiempos de entrega de pedidos fuera del puro ámbito empírico resulta de cabal importancia considerando la fuerte competitividad entre las distintas empresas del área. Conocer exactamente el tiempo que demoran los procesos se traduce directamente en una ventaja competitiva y, sobre todo, a nivel productivo interno. Lo segundo, porque se le otorga a la empresa un conocimiento real de cómo medir la productividad de su industria, permitiéndole hacer un análisis de porqué demora más del tiempo esperado la terminación de un pedido.
Como se mencionaba, la ventaja competitiva que otorga el conocimiento de tiempos de entrega exactos permite conocer si vale la pena la inversión en nueva maquinaria o el ajuste productivo de los empleados. Además, al negociar los tiempos de entrega con los clientes, esto le permite tener más días de gracia por cualquier eventualidad en el proceso.
Por eso la importancia de este estudio, el cual ciertamente serviría para industrias manufactureras pequeñas, o en vías de crecimiento. Sin embargo, se plantean heurísticas que podrían utilizarse en caso de un mayor número de máquinas para las industrias más grandes. Además, se podrían añadir y evaluar el modelo con nuevas restricciones para las industrias alimenticias. Para nuestro caso, la empresa Bórdalo, ubicada en la ciudad de Guayaquil, una manufacturera pequeña se ve beneficiada por el modelo propuesto, al tener un número pequeño de máquinas a su disposición. Dicha empresa se especializa en bordados de camisetas, tanto de nombres como de logotipos para personas naturales y sobre todo para empresas que requieren uniformes para su personal. Además del bordado de logotipos o nombres, la empresa está iniciando su ampliación a confección de camisetas disponibles para la venta con o sin bordado.
La empresa Bórdalo de Guayaquil, Ecuador, se comprometió a permitir el estudio del problema planteado. Para ello, se tomarán los datos de tiempos de procesamiento individuales de las máquinas en los distintos productos de interés. Para la selec-

ción de los productos de interés, se considerará su volumen de producción.
Una vez tomados los datos se construirá la matriz bajo la cual se construirá el programa de optimización lineal en un lenguaje de modelización matemática, cuyo código será realizado oportunamente y con base en los estudios y los objetivos mencionados anteriormente. Una vez obtenida la solución del programa lineal, se realizará un análisis de los mismos. Y, de ser el caso, se realizarán simulaciones que permitan analizar con mayor profundidad los resultados obtenidos. Dichas simulaciones también permitirán hacer un breve análisis de la conveniencia de adquirir o no nueva maquinaria o de renovar maquinaria antigua en caso de sugerirlo el análisis descrito.
Para comparar los resultados obtenidos por el programa lineal, se plantearán heurísticas de solución al problema. Adicionalmente se realizarán pruebas computacionales con instancias simuladas de mayor tamaño.

1.2. Objetivos

El objetivo principal planteado para el presente trabajo es *formular y resolver un modelo de programación lineal entera asociado al problema de minimizar el tiempo total que tarda la empresa manufacturera Bórdalo en culminar sus pedidos usando métodos exactos y heurísticos*.
En términos específicos, se desea

- Estimar los tiempos de entrega de encargos por máquina.
- Construir un modelo de programación lineal que represente el caso de estudio.
- Resolver el problema planteado mediante el uso de un solver de programación lineal o un algoritmo heurístico.
- Comparar la solución dada por la heurística con el modelo de programación lineal.
- Proponer una solución factible para el problema en el caso de n productos y m máquinas.

1.3. Estado del arte

Gupta y Dudek (1971) y Gary et al. (1995) describen el conflicto existente entre criterios óptimos para secuencias de trabajos y los objetivos corporativos como, por ejemplo, hacer énfasis en la producción descuidando el servicio y atención al cliente que permite priorizar la importancia de los pedidos. Sin embargo, al no tener en cuenta las capacidades individuales de las máquinas y optimizar su producción, la priorización de los pedidos puede poco añadir a establecer una organización de la producción de los pedidos, pues puede darse el caso en el que realmente no sea requerida una priorización si se optimizan previamente los tiempos de producción. Por eso resulta necesario, como primera instancia, establecer parámetros que permitan optimizar el makespan en un conjunto de trabajos para un grupo de máquinas. El *makespan* se define como el tiempo máximo para completar un conjunto de trabajos en un grupo de máquinas. Puede ser visto como el tiempo necesario para terminar todo un plan de producción pues se mide desde el comienzo del primer trabajo hasta la finalización del último.
El problema **P1** descrito, es demostrado como NP-duro (Garey y Johnson, 1979) en general; y, en particular, para dos máquinas también está demostrado ser NP-duro (Lenstra, 1977). Usando dos máquinas idénticas trabajando paralelamente, Lenstra (1990) demuestra que un algoritmo polinomial para el problema general, en el peor de los casos, tiene una aproximación a la solución óptima a razón de $3/2$. Este trabajo se basa, sobre todo, en el trabajo realizado por Fanjul-Peyró y Ruiz (2010), quienes encontraron resultados alentadores junto con Vallada y Ruiz (2010) usando un solver de IBM CPLEX 11.0. El problema estudiado a profundidad por Fanjul-Peyró y Ruiz (2010) estaba definido solamente con las sigueintes restricciones:

- Cada trabajo es asignado exactamente una máquina.
- El *makespan* es igual o mayor que la suma de los tiempos de procesamiento de los trabajos asignados a cada máquina.

El problema puede ser extendido satisfactoriamente a una gran cantidad de máquinas (Fanjul-Peyró y Ruiz, 2010), a pesar de ser un problema NP-duro por extensión. Para este caso, usando CPLEX 11.0, los resultados óptimos se hallaron para problemas con ocho trabajos y cinco máquinas y solamente en algunos casos para algunos problemas de hasta doce trabajos (Vallada y Ruiz, 2010), es decir, para problemas muy pequeños.
Con base en los métodos de programación lineal entera mixta propuestos por Ruiz

(2010), se establece un modelo de programación entera que permite hallar la solución al problema en estudio. Se añaden restricciones que adecuan el problema en estudio al programa lineal. Por ejemplo, en nuestro caso el problema exige establecer procesos por encima de máquinas. Es decir, un pedido consiste en varios procesos y uno de estos procesos puede ser realizado por más de una máquina. De modo que el trabajo debe pasar siempre por los procesos necesarios, pero no necesariamente por todas las máquinas. En este sentido se añaden restricciones con respecto a esta secuencia particular.
Además, al ser un problema NP-duro (Garey y Johnson, 1979), se buscarán heurísticas de solución al problema. Estas últimas, basadas en la heurística NEH (Nawaz-Enscore-Ham) de Nawaz et al. (1983), la cual es la más comúnmente implementada debido a su eficiencia. A pesar del uso de permutaciones en el algoritmo, mantiene su eficiencia frente a los modelos lineales.

1.4. Programación dinámica del proceso de trabajo

Usando un solver de programación lineal, y considerando los resultados obtenidos por Vallada y Ruiz (2010), se resolverá el modelo para casos pequeños, es decir una empresa en la industria manufacturera que no sobrepase de las 5 máquinas y no más de ocho trabajos, intentado así obtener una solución óptima del problema. Es importante, sin embargo, simular los resultados aumentando el número de máquinas y trabajos para poder comparar los resultados obtenidos y analizar el alcance del modelo para industrias con mayor cantidad de máquinas y trabajos. Además, al ser un problema NP-duro (Garey y Johnson, 1979), se buscaron heurísticas de solución al problema.

1.5. Estructura del trabajo

A continuación, se presentará, con más profundidad, la teoría sobre la cual se desarrolló el modelo en el Capítulo 2, y se revisarán brevemente distintos modelos que se han desarrollado en los últimos años en torno al problema general que se estudia; en el Capítulo 3, se particulariza el problema de organización de trabajos (*Job Shop Scheduling*) para las máquinas con el fin de estimar sus tiempos de entrega. Se presentan, asimismo, distintos modelos y sus aplicaciones brevemente; el Capí-

tulo 4 recoge el trabajo práctico realizado: la selección del modelo para el caso de estudio y los resultados obtenidos. Además, se presentan simulaciones y pruebas computacionales de las heurísticas implementadas para resolver el problema real y los problemas simulados; fnalmente, se presentan una serie de conclusiones y recomendaciones que se considera que deberían analizarse para mejorar el modelo o para extenderlo como se ha mencionado en párrafos anteriores.

Capítulo 2

Job Shop Scheduling Problem

2.1. Introducción

Un *schedule* es una asignación específica de tareas u operaciones a los recursos en un tiempo determinado. El objetivo de este tipo de problemas recae sobre la minimización del *makespan*, que es el tiempo en el que se termina el último trabajo. La programación de las tareas a realizarse en una industria manufacturera debe considerar, no solo las limitaciones de las máquinas, sino también las limitaciones humanas en la operación de las mismas, para poder determinar un tiempo real de finalización del *schedule*. También se tiene que, según el tipo de trabajo que se desee realizar, las máquinas varían en sus rendimientos. Es decir, una máquina puede tardar más en realizar determinado trabajo que otra, mientras que puede tardar menos en realizar un distinto trabajo que otra máquina.
Un modelo de organización de trabajos (*Job Shop Scheduling*) es una abstracción del problema de decisión de un programa laboral (*schedule*), en el cual se considera un sistema de operaciones o trabajos, las restricciones en el procesamiento y otros criterios.

2.2. Terminología

Se considera un *scheduling model* determinado por un número de trabajos que deben ser procesados por un número definido de máquinas. Específicamente, se asume un conjunto N de trabajos, indexados $N = \{1, 2, \ldots, n\}$. Los subíndices j y k serán utilizados para referirse a trabajos del conjunto N.

Similarmente, se tiene un conjunto M de m máquinas. El subíndice i se referirá a cualquier máquina del conjunto M. Las máquinas y los trabajos serán independientes y las instancias se considerarán determinísticas y conocidas a priori. A lo largo del documento, se trabajará con los siguientes términos:

- x_{ijk}**:** variable binaria que indica si el trabajo j precede al trabajo k en la máquina i.
- c_{ij}**:** tiempo que cuesta realizar el trabajo j en la máquina i.
- s_{ijk}**:** es el tiempo que tarda el trabajo j en pasar al trabajo k en la máquina i.
- p_{ij}**:** tiempo que tarda la máquina i en realizar el trabajo j.
- $C_{\text{máx}}$**:** es el *makespan*, es decir, el tiempo total de finalización de todos los trabajos.

2.3. Modelo

Muchos modelos de programación de máquinas manufactureras pueden ser formulados como programas enteros o binarios, de modo que se pueden resolver a través de distintos métodos de *branch and bound*. El método consiste básicamente en dos pasos:

1. Formulación de la programación de la fabricación a través de un modelo de Programación Lineal Entera Mixta, MILP por sus siglas en inglés. Esta formulación puede: directamente dar la solución o referir a un modelo intermedio de programación entera, como por ejemplo, modelos de la teoría de grafos.
2. Aplicar el software de programación entera al modelo MILP generado. Aquí es donde se ejecutan los ajustes de los parámetros para asegurar la efectividad del modelo: márgenes de error, estrategias de ramificación, etc.

Previo a introducir el modelo que se usará en este trabajo, se presenta el modelo general sobre el cual se basa el modelo final trabajado. Considérense modelos de programación con dos máquinas paralelas. Se asume lo siguiente:

- Cada trabajo j, $j = 1, \ldots, n$ tiene que ser procesado por exactamente una de las m máquinas.

- Ninguna máquina puede procesar más de un trabajo al mismo tiempo. Es más, cuando un trabajo está siendo ejecutado por una máquina, esta continuará hasta que termine el trabajo.

- El tiempo de procesamiento de un trabajo es conocido y finito positivo. En general, cada máquina i ($i = 1, \ldots, m$) tiene un tiempo de procesamiento $p_{ij} > 0$ distinto al procesar el trabajo j.

Denótese este modelo como $R \mid\mid C_{\text{máx}}$. Nótese que, en realidad, este es un problema de asignación pues el orden de procesamiento de los trabajos asignados a una máquina no altera el tiempo máximo de finalización de esa máquina. De modo que existen m^n soluciones posibles al problema.
El problema $R \mid\mid C_{max}$ es NP-duro debido a que el caso particular con máquinas idénticas ($P \mid\mid C_{max}$) fue demostrado ser NP-duro por Garey y Johnson (1979). Inclusive la versión de 2 máquinas ($P2 \mid\mid C_{max}$) es NP-duro de acuerdo a Lenstra (1977). Se puede ver que, aún el problema con dos máquinas idénticas es difícil de resolver. Es más, Lenstra (1990) demostró que no existe un algoritmo de tiempo polinomial para el problema $R \mid\mid C_{max}$ con un radio de aproximación mejor que $3/2$.
Un modelo MILP para $R \mid\mid C_{max}$ contiene las siguientes variables:

$$\begin{aligned} x_{ij} &= \begin{cases} 1 & \text{si el trabajo } j \text{ es asignado a la máquina } i \\ 0 & \text{caso contrario} \end{cases} \\ C_{\text{máx}} &= \textit{makespan} \end{aligned}$$

La función objetivo es la minimización del *makespan*:

$$\text{mín } C_{\text{máx}}$$

Y las restricciones:

$$\begin{aligned} &\textstyle\sum_{i=1}^{m} x_{ij} = 1 && \forall j \in N \\ &\textstyle\sum_{j=1}^{n} p_{ij} \cdot x_{ij} \leq C_{\text{máx}} && \forall i \in M \\ &x_{ij} \in \{0, 1\} && \forall j \in N, \forall i \in M \end{aligned}$$

La primera restricción asegura que cada trabajo sea asignado a exactamente 1 máquina. La segunda simplemente determina que el *makespan* es mayor o igual a la suma de los tiempos de procesamiento asignados a cada máquina. Esto se repite para cada máquina porque, en este problema, el *makespan* está dado por la máquina que más se tarde en acabar sus trabajos.
Ahora, se procede a añadir al modelo restricciones de secuencia de los trabajos en

las máquinas, es decir, que cada trabajo requiere de la intervención de más de una máquina. Para aclarar este punto, se va a hablar de procesos, trabajos y máquinas. Un trabajo conlleva varios procesos y cada proceso puede ser realizado por varias máquinas. De modo que, las restricciones de secuencia implica que, para realizar el trabajo, el producto debe pasar consecutivamente por varios procesos. Es decir, un producto no puede comenzar el proceso 2 sin haber terminado el proceso 1.
Denótese la variable s_{ijk} correspondiente a la secuencia de procesos que debe seguir un trabajo k en la máquina i después de haber procesado el trabajo j.
Una de las implicaciones de ingresar esta nueva variable es que el modelo anterior deja de ser un modelo de asignación y, dependiendo de cuán diferentes sean las secuencias de los trabajos asignados a su vez a las máquinas, el tiempo de finalización del trabajo varía. El nuevo modelo MILP tiene las siguientes variables de decisión:

$$\begin{aligned} x_{ijk} &= \begin{cases} 1 & \text{si el trabajo } j \text{ precede al trabajo } k \text{ en la máquina } i \\ 0 & \text{caso contrario} \end{cases} \\ c_{ij} &= \text{Tiempo de finalización del trabajo } j \text{ en la máquina } i \\ C_{\text{máx}} &= \text{makespan} \end{aligned}$$

Nótese que hay $n(n-1)m$ variables binarias, las cuales crecen a medida que crecen la cantidad de trabajos y máquinas consideradas en una instancia. La variable X_{ijj} no está definida, sin embargo, el número de variables crece debido a que se requieren de variables *dummy*. El detalle se puede ver a continuación:
Función objetivo:

$$\text{mín } C_{\text{máx}}$$

Y las restricciones son:

$$\sum_{i \in M} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \quad \forall k \in N \tag{2.1}$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} x_{ijk} \leq 1 \quad \forall j \in N \tag{2.2}$$

$$\sum_{k \in N} x_{i0k} \leq 1 \quad \forall i \in M \tag{2.3}$$

$$\sum_{\substack{h \in \{0\} \cup \{N\} \\ h \neq k, h \neq j}} x_{ihj} \geq x_{ijk} \quad \forall j, k \in N, j \neq k, \forall i \in M \tag{2.4}$$

$$C_{ik} + V(1 - x_{ijk}) \geq C_{ij} + s_{ijk} + p_{ik} \quad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \tag{2.5}$$

$$C_{i0} = 0 \quad \forall i \in M \tag{2.6}$$

$$C_{ij} \geq 0 \quad \forall j \in N, \forall i \in M \tag{2.7}$$

$$C_{ij} \leq C_{\text{máx}} \quad \forall j \in N, \forall i \in M \tag{2.8}$$

$$x_{ijk} \in \{0,1\} \quad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \tag{2.9}$$

donde

2.1: asegura que cada trabajo es asignado exactamente a una máquina y tiene exactamente un predecesor. El conjunto sobre el que se realiza la sumatoria de los j trabajos, $\{0\} \cup \{N\}$, incluye trabajos *dummies* creados para especificar el predecesor de cada una de las máquinas. En caso de que el trabajo es el primero, los trabajos *dummy* permiten ejecutar el programa.

2.2: se determina el número de sucesores de cada trabajo a un máximo de 1.

2.3: limita el número de sucesores de un trabajo *dummy* a un máximo de 1 por cada máquina.

2.4: asegura que los trabajos están propiamente asignados a una máquina, es decir, si un trabajo j es procesado en una máquina i, un predecesor h debe existir en la misma máquina.

2.5: controla el tiempo de finalización de los trabajos en las máquinas. Si un trabajo k es asignado a una máquina i luego del trabajo j (i.e. $x_{ijk} = 1$), su tiempo de finalización c_{ik} debe ser mayor que el tiempo de finalización de j: c_{ij} más el tiempo de procesamiento del trabajo k.

2.6: define tiempos de finalización como 0 para trabajos $dummy$.

2.7: define tiempos de finalización como no negativos para trabajos regulares.

2.8: define el *makespan*.

2.9: define las variables binarias.

Este modelo, de acuerdo con Vallada y Ruiz (2010), probado en CPLEX 11.0, encuentra soluciones óptimas hasta para 8 trabajos y 5 máquinas. Aunque también logra alcanzar optimalidad para instancias con 10 y 12 trabajos. Nótese que las instancias son muy pequeñas, por lo que, para instancias mayores, se requerirán de heurísticas.

Denótese este modelo como **M1**. Con base en este modelo, en el siguiente capítulo se detallarán los cambios que se realizaron para adecuarlo al problema particular de este estudio.

Capítulo 3

Estimación de tiempos de entrega

En este capítulo, se ahonda en el modelo y las heurísticas implementadas con las cuales se trabajaron para obtener los resultados. Como se mencionó en el capítulo anterior, se realizaron pequeñas modificaciones al modelo **M1** para adecuarlo a las necesidades de la empresa.

3.1. Introducción

El modelo que se pretende conseguir con la empresa, no consiste solamente en la asignación sencilla de trabajos a máquinas, sino que el problema es como se describe a continuación:

- Un pedido consiste en uno, dos y hasta tres procesos:
  - solo confección de camisetas.
  - confección y bordado: la manufactura de las camisetas y el bordado.
  - diseño y bordado: se recepta el logo en un formato específico y este tiene que ser adaptado para el programa que maneja las máquinas. Luego se borda en la prenda o tela (servilletas, manteles, banderas, etc.).
  - confección, diseño y bordado.
- El trabajo en cada proceso varía en su tiempo no solo por la capacidad de la máquina sino por el tipo de producto:
  - la confección varía según si la camiseta es con cuello o con cuello redondo o con cuello en "v".

- en los bordados no varía solamente en gorras, pues se requiere de un instrumento especial para las mismas, que delimita la cantidad de puntadas del bordado. Sin embargo, para las demás prendas, el tiempo varía según la cantidad de puntadas que requiere el bordado.
- en el diseño el tiempo varía según la complejidad del logo.

- Para cada proceso se tienen varias máquinas, lo cual implica una nueva restricción, pues en el modelo **M1**, el producto debe recorrer todas las máquinas, ya que cada máquina implica un proceso.

El estudio presente no consiste en la evaluación de inventarios, que correspondería al detalle de cómo mantener en bodega las camisetas confeccionadas según la demanda. Se entiende entonces que la confección de camisetas no forma parte de este modelo en cuanto que no se realiza in situ con el pedido, sino que se suelen tener ya confeccionadas con tiempo y a la venta. Sin embargo, el diseño y bordado de las prendas requiere siempre de una optimización en cuanto a la urgencia de los pedidos y el uso de las capacidades individuales de las máquinas.

3.2. Modelo

El modelo implementado para obtención de resultados considera las siguientes variables:

$$x_{ijk} = \begin{cases} 1, & \text{si el trabajo j precede al trabajo k en la máquina i} \\ 0, & \text{caso contrario} \end{cases}$$

$$c_{ij} = \text{Tiempo en el cuál se completó el trabajo j en la máquina i}$$

$$C_{max} = \text{Tiempo total en el cuál se terminan todos los trabajos}$$

La función objetivo:

$$\text{mín}\, c_{max}$$

Y las restricciones:

$$\sum_{i \in M} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \qquad \forall k \in N \tag{3.1}$$

$$\sum_{i \in M1} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \qquad \forall k \in N \text{ con k par} \tag{3.2}$$

$$\sum_{i \in M2} \sum_{\substack{j \in \{0\} \cup \{N\} \\ j \neq k}} x_{ijk} = 1 \qquad \forall k \in N \text{ con k impar} \tag{3.3}$$

$$\sum_{i \in M} \sum_{\substack{k \in N \\ j \neq k}} x_{ijk} \leq 1 \qquad \forall j \in N \tag{3.4}$$

$$\sum_{k \in N} x_{i0k} \leq 1 \qquad \forall i \in M \tag{3.5}$$

$$\sum_{i \in M} c_{ij} + \sum_{\substack{l \in \{0\} \cup \{N\} \\ l \neq j+1}} x_{kl(k+1)} p_{k(j+1)} \leq c_{k(j+1)} \qquad \forall k \in M, \forall j \in N \text{ con j impar} \tag{3.6}$$

$$\sum_{\substack{h \in \{0\} \cup \{N\} \\ h \neq k, h \neq j}} x_{ihj} \geq x_{ijk} \qquad \forall j, k \in N, j \neq k, \forall i \in M \tag{3.7}$$

$$c_{ik} + V(1 - x_{ijk}) \geq c_{ij} + s_{ijk} + p_{ik} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \tag{3.8}$$

$$c_{io} = 0 \qquad \forall i \in M \tag{3.9}$$

$$c_{ij} \geq 0 \qquad \forall j \in N, \forall i \in M \tag{3.10}$$

$$c_{ij} \leq c_{max} \qquad \forall j \in N, \forall i \in M \tag{3.11}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall j \in \{0\} \cup \{N\}, \forall k \in N, j \neq k, \forall i \in M \tag{3.12}$$

donde $M = M1 \cup M2$ y:

3.1: asegura que cada trabajo es asignado exactamente a una máquina y tiene exactamente un predecesor.

3.2: asegura que cada trabajo es asignado exactamente a una máquina y tiene exactamente un predecesor considerando solamente el grupo de máquinas del 1er proceso, donde se consideran los trabajos *pares*.

3.3: como en (3.2), pero para el caso de los trabajos *impares* o del 2do proceso.

3.4: se determina el número de sucesores de cada trabajo a un máximo de 1.

3.5: limita el número de sucesores de un trabajo *dummy* a un máximo de 1 por cada máquina.

3.6: controla que cada máquina complete los dos procesos.
3.7: asegura que los trabajos están propiamente asignados a una máquina, es decir, si un trabajo j es procesado en una máquina i, un predecesor h debe existir en la misma máquina.
3.8: controla el tiempo de finalización de los trabajos en las máquinas. Si un trabajo k es asignado a una máquina i luego del trabajo j (i.e. $X_{ijk} = 1$), su tiempo de finalización C_{ik} debe ser mayor que el tiempo de finalización de j: C_{ij} más el tiempo de procesamiento del trabajo k.
3.9: define tiempos de finalización como 0 para trabajos $dummy$.
3.10: define tiempos de finalización como no negativos para trabajos regulares.
3.11: define el *makespan*.
3.12: define las variables binarias.

Nótese que al modelo **M1** presentado al final del capítulo 2, se le añaden únicamente las restricciones **3.2**, **3.3** y **3.6**. Estas restricciones satisfacen las necesidades particulares del problema en estudio mencionadas anteriormente.

3.3. Heurísticas

El algoritmo se trabajó con base en el NEH de Nawatz el al. (1983). Está heurística resuelve el problema con permutaciones de los trabajos, ordenándolos según un índice predefinido. El procedimiento de NEH se basa en la idea de que los trabajos con la mayor suma en tiempos de procesamiento en todas las máquinas debe ser comenzado cuanto antes. El algoritmo es el siguiente:

Algorithm 1 NEH

1: **Input:** Datos de la instancia
2: **Salida:** Secuencia de trabajos Π, Makespan C_{max}
3: Sea $P_j = \sum_{i=1}^{m} p_{ij}$
4: Sea $\Pi = \varnothing$, $J = 1, \ldots, n$, verificando $P_1 \geq P_2 \geq \ldots P_n$
5: Sea Π la mejor permutación entre (1,2) y (2,1)
6: **for** $k = 3$ a n **do**
7: Insertar el trabajo k en la posición de Π que mantenga el menor *makespan*
8: **Return:** Π, $C_{\text{máx}}$

Así, por ejemplo, se pueden ver en la Figura 3.1 cómo funcionan las iteraciones del algoritmo NEH, intercambiando el orden de los trabajos buscando siempre un menor *makespan* en cada intercambio:

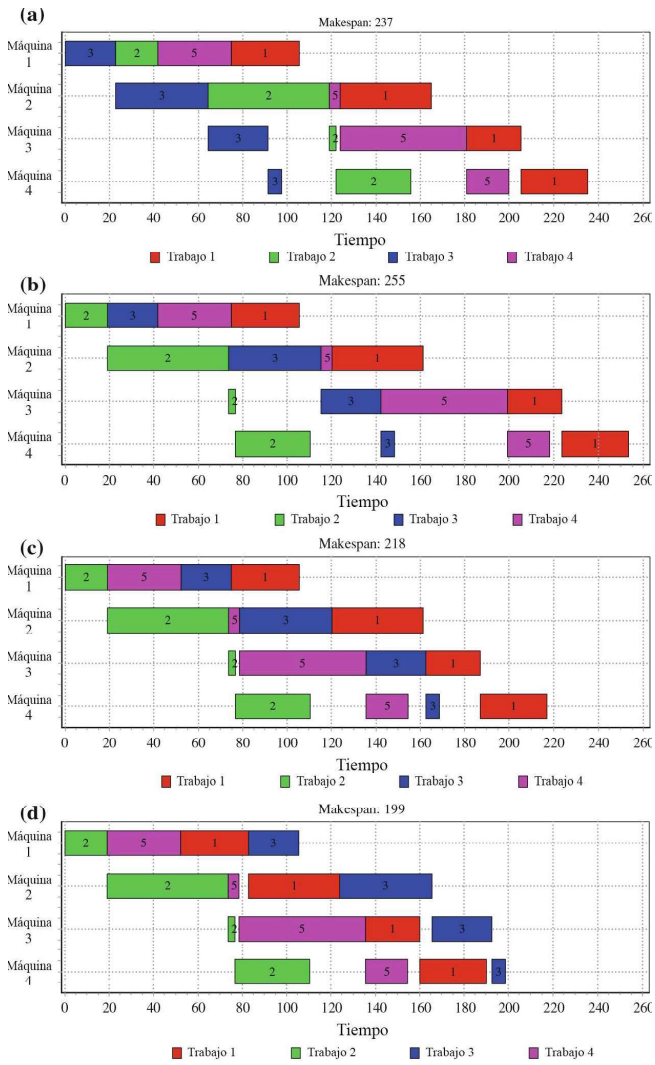

Figura 3.1: Iteraciones del algoritmo NEH

En el caso de estudio, se modificó el algoritmo NEH pues los trabajos no necesariamente recorren todas las máquinas, aunque sí todos los procesos. Para ello, se construyó el algoritmo 3.

En el algoritmo se puede ver que, previo a la verificación de todos los posibles ordenamientos de los trabajos, se realiza un ordenamiento según los tiempos tal como se realiza en el NEH. Es decir, se ordena de mayor a menor todos los trabajos según el tiempo que demoran. Luego, al fijarse el trabajo que más tiempo toma en realizarse como el primero, se procede a verificar todos los posibles ordenamientos que puede seguir el esquema de trabajo.

Además de la heurística descrita, se le añadió un cambio de ordenamiento alea-

Algorithm 2 Heurística Base

Entrada: Datos de la instancia
Salida: Secuencia de trabajos Π, Makespan C_{max}
Sea $T1 = \{1, ..., k_1\}$ los trabajos del primer proceso
Sea $T2 = \{1, ..., k_2\}$ los trabajos del segundo proceso, donde el trabajo n-ésimo sucede al trabajo n-ésimo de T1
Sea $M1 = \{1, ..., m_1\}$ las máquinas que realizan el primer proceso
Sea $M2 = \{1, ..., m_2\}$ las máquinas que realizan el segundo proceso
$t_{i,M1} = \text{máx}_{j \in M1}\, p_{i,j}, i \in T1$
$t_{i,M2} = \text{máx}_{j \in M2}\, p_{i,j}, i \in T2$
$tiempo_i = t_{i,M1} + t_{j,M2}, i \in T1, j \in T2$ tal que j es el trabajo sucesor de i
Se define un orden de entrada de trabajos del conjunto T1 como $U = 1, ..., k_1$ en orden descendente de acuerdo con los valores obtenidos en *tiempo*
for i en el conjunto de trabajos U **do**
    makespan := Un valor lo suficientemente grande.
    R := El conjunto de posiciones posibles sobre las maquinas M1.
    S := El conjunto de posiciones posibles sobre las máquinas M2.
    for m en el conjunto R **do**
        for n en el conjunto S **do**
            Ingresar el trabajo i en la posición m sobre las máquinas M1 y el trabajo predecesor de i del conjunto T2 en la posición n sobre las máquinas M2
            makespanParcial := tiempo total de finalización incluido el trabajo recién ingresado
            if $makespan > makespanParcial$ **then**
                $makespan = makespanParcial$
                Almacenar la posición m, n como posible solución para el trabajo i

torio a la asignación inicial de los trabajos en las máquinas. Es decir, en lugar del décimo paso, donde se define un orden de ingreso de los trabajos según su tiempo, se establece un orden aleatorio. Con esta modificación en el algoritmo, se verá más adelante, se obtuvieron mejores resultados. Esta heurística se denota como Heurística OA (Orden Aleatorio) y su pseudocógido se presenta a continuación:

Algorithm 3 Heurística Ordenamiento Aleatorio

Entrada: Datos de la instancia
Salida: Secuencia de trabajos Π, Makespan C_{max}
Sea $T1 = \{1, ..., k_1\}$ los trabajos del primer proceso
Sea $T2 = \{1, ..., k_2\}$ los trabajos del segundo proceso, donde el trabajo n-ésimo sucede al trabajo n-ésimo de T1
Sea $M1 = \{1, ..., m_1\}$ las máquinas que realizan el primer proceso
Sea $M2 = \{1, ..., m_2\}$ las máquinas que realizan el segundo proceso
Sea V la cantidad de veces que se recorre el Algoritmo 2

for v en V **do**
    Se define un orden de entrada de trabajos de acuerdo con un ordenamiento aleatorio de los trabajos
    for i en el conjunto de trabajos U **do**
        makespan := Un valor lo suficientemente grande.
        R := El conjunto de posiciones posibles sobre las maquinas M1.
        S := El conjunto de posiciones posibles sobre las máquinas M2.
        for m en el conjunto R **do**
            for n en el conjunto S **do**
                Ingresar el trabajo i en la posición m sobre las máquinas M1 y el trabajo predecesor de i del conjunto T2 en la posición n sobre las máquinas M2
                makespanParcial := tiempo total de finalización incluido el trabajo recién ingresado
                if $makespan > makespanParcial$ **then**
                $makespan = makespanParcial$
                Almacenar la posición m, n como posible solución para el trabajo i
        Guarda el *schedule* con el menor *makespan*

Capítulo 4

Aplicación del modelo

En este capítulo se detallan los resultados obtenidos de la implementación del modelo y las heurísticas. Además se presentan, en primer lugar, los datos obtenidos de la empresa con los cuales se trabajaron los modelos.

Luego se presentan los diferentes resultados, tanto de simulaciones como de los datos de la empresa en tablas y, visualmente, con diagramas de Gantt. Estos diagramas están diseñados para programar actividades en el tiempo, por lo que resultan adecuados para presentar los resultados obtenidos.

4.1. Sobre los datos

La empresa permitió la revisión de su base de datos de tiempos de ejecución de trabajos. Ciertamente estos tiempos fueron tomados por la empresa para ayudar al estudio, por lo que se les solicitaron 30 mediciones para cada agrupación que se consideró conveniente. Para la implementación del modelo, los datos fueron agrupados en 8 categorías:

- bordados en gorras con menos de 2000 puntadas
- bordados en gorras con más de 2000 puntadas
- bordados en prendas con menos de 2000 puntadas
- bordados en prendas entre 2000 y 4000 puntadas
- bordados en prendas entre 4000 y 6000 puntadas
- bordados en prendas entre 6000 y 8000 puntadas

- bordados en prendas entre 8000 y 10000 puntadas
- bordados en prendas con más de 10000 puntadas

Entiéndase que las puntadas son realmente las que definen el tamaño del bordado, por lo que, según la experiencia de la empresa y el uso de aros de fijación de bordados se definieron las categorías previamente. Cabe notar que la toma de los datos en gorras y hasta la categoría más baja de bordados medianos resultaron relativamente sencillos de conseguir debido a su frecuencia de pedidos. Sin embargo, tomó mucho más tiempo conseguir los tiempos de bordados de más de 9000 puntadas.

En el Cuadro 4.1 se presenta un ejemplo de los datos obtenidos de la empresa.

Cuadro 4.1: Tiempos de bordados en gorras en minutos

	Bordado en gorras	
	Menos de 3000 puntadas	**Más de 3000 puntadas**
1	4,79	7,22
2	4,77	8,04
3	4,76	7,91
4	4,76	9,18
5	4,80	9,42

En el anexo se pueden ver los tiempos tomados para la extracción del promedio de tiempos con los cuales se trabajaron el modelo y las heurísticas.

Los datos obtenidos de la empresa siguen una distribución normal. En el apéndice se puede ver todas las pruebas de hipótesis realizadas para cada grupo de datos. Es importante que los datos sean normales pues, por el teorema central del límite, se puede trabajar con el promedio como un valor válido para la tabla de tiempos inicial al implementar el modelo y las heurísticas.

A modo de ejemplo, se presenta el histograma para los tiempos en bordados pequeños con menos de 3.000 puntadas en la Figura 4.1.

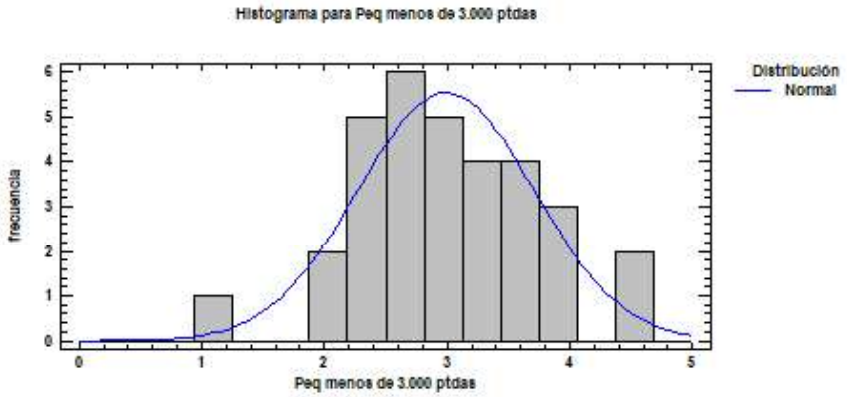

Figura 4.1: Histograma para tiempos de bordados pequeños con menos de 3.000 puntadas. Realizado en Statgraphics.

Se realizó una prueba de normalidad de los datos con el estadístico W de Shapiro-Wilk. El estadístico obtuvo un valor de $0,977939$ y su respectivo valor-P $0,78072$. Al ser el valor-P mayor que $0,05$, no se puede rechazar la idea de que los datos no sigan una distribución normal.

Para la sección siguiente conviene conocer que se utilizó una máquinas con las siguientes especificaciones:

- **OS:** Ubuntu 18.04.2 LTS x86_64
- **CPU:** Intel i3 M330 (4) @ 2.133GHz
- **GPU:** NVIDIA GeForce 310M
- **Memoria:** 2168MiB / 3862MiB

Hay que considerar que las especifaciones afectan a los tiempos de simulación que se reportan en este trabajo.

En cuanto a la programación de los modelos, se realizón en Python en la distribución Anaconda, la cual sincroniza directamente con Jupyter Lab, donde se hicieron los gráficos. Los paquetes utilizados fueron:

- numpy,
- pandas,
- operator,

- copy y
- timeit

Para los modelos lineales se utilizó el programa Gurobi. En los anexos se encuentran los códigos implementados tanto para las heurísticas como para el modelo lineal.

4.2. Simulaciones

A continuación, se presentan resultados de simulaciones realizadas con base en las instancias descritas en el Cuadro 4.2.

Cuadro 4.2: Instancias

Instancia	**M1**	**M2**	**Número de trabajos**
1	2	2	10
2	2	2	16
3	3	2	10
4	3	2	12
⋮	⋮	⋮	⋮
9	4	3	10
10	4	3	12
11	4	3	14
12	4	3	16
13	4	4	12
14	4	4	16
⋮	⋮	⋮	⋮
21	6	6	50

La columna **instancia** se refiere a un id de instancia que servirá para visualizar gráficamente los resultados más adelante. **M1** y **M2** se refiere a los conjuntos de máquinas disponibles para cada uno de los procesos. Así, en la instancia 9, se tienen

4 máquinas para el primer proceso y 3 máquinas para el segundo proceso. Y las simulaciones se realizaron para 10 trabajos pendientes.

Cuadro 4.3: Resultados de tiempos de ejecución, gap y *makespan* de las simulaciones para el modelo lineal y la heurística base

	Modelo Lineal			Heurística Base		
Instancia	**t (seg)**	**gap**	**makespan**	**t (seg)**	**makespan**	**gap**
1	2.3	0.0 %	23.2	0.0	25.1	8.2 %
2	277.4	0.0 %	31.0	0.1	33.2	7.1 %
3	1.3	0.0 %	17.1	0.0	20.8	21.6 %
4	76.2	0.0 %	17.0	0.1	20.6	21.2 %
5	2.0	0.0 %	15.8	0.0	18.4	16.5 %
6	3.2	0.0 %	15.8	0.1	16.4	3.8 %
7	6.6	0.0 %	16.8	0.1	17.1	1.8 %
8	150.8	0.0 %	21.9	0.3	24.0	9.6 %
9	0.6	0.0 %	14.0	0.1	17.1	22.1 %
10	2.6	0.0 %	16.8	0.1	19.1	13.7 %
11	4.0	0.0 %	14.2	0.1	14.2	0.0 %
12	30.8	0.0 %	16.1	0.2	16.6	3.1 %
13	1.7	0.0 %	14.3	0.1	15.3	7.0 %
14	23.3	0.0 %	13.2	0.2	15.3	15.9 %
15	449.1	0.0 %	13.7	0.4	16.7	21.9 %
16	1.8	0.0 %	11.8	0.1	13.5	14.4 %
17	103.4	0.0 %	13.7	0.4	14.9	8.8 %
18	1,089.4	33.1 %	16.0	1.5	16.8	
19	1,588.7	79.6 %	49.0	8.2	32.8	
20	2,457.6	91.0 %	74.8	45.3	52.7	
21	59.9	0.0 %	10.9	0.5	12.4	13.8 %
22	1,869.6	31.7 %	15.8	1.7	17.6	
23	2,567.6	91.4 %	104.4	104.2	47.9	
24	3,020.4	77.5 %	37.8	9.4	19.8	

Cuadro 4.4: Resultados de tiempos de ejecución, gap y *makespan* de las simulaciones para el modelo lineal y la heurística OA

	Modelo Lineal			Heurística OA			
Instancia	**t (seg)**	**gap**	**makespan**	**t (seg)**	**makespan**	**n**	**gap**
1	2.3	0.0 %	23.2	1.3	23.2	50.0	0.0 %
2	277.4	0.0 %	31.0	5.9	31.0	50.0	0.0 %
3	1.3	0.0 %	17.1	1.6	17.1	50.0	0.0 %
4	76.2	0.0 %	17.0	3.3	17.0	50.0	0.0 %
5	2.0	0.0 %	15.8	1.9	15.8	50.0	0.0 %
6	3.2	0.0 %	15.8	3.2	15.8	50.0	0.0 %
7	6.6	0.0 %	16.8	5.0	16.8	50.0	0.0 %
8	150.8	0.0 %	21.9	7.3	21.9	50.0	0.0 %
9	0.6	0.0 %	14.0	2.2	14.0	50.0	0.0 %
10	2.6	0.0 %	16.8	3.6	16.8	50.0	0.0 %
11	4.0	0.0 %	14.2	5.6	14.2	50.0	0.0 %
12	30.8	0.0 %	16.1	8.4	16.1	50.0	0.0 %
13	1.7	0.0 %	14.3	4.2	14.3	50.0	0.0 %
14	23.3	0.0 %	13.2	9.3	13.5	50.0	2.3 %
15	449.1	0.0 %	13.7	18.6	14.3	50.0	4.4 %
16	1.8	0.0 %	11.8	4.8	11.8	50.0	0.0 %
17	103.4	0.0 %	13.7	20.3	13.7	50.0	0.0 %
18	1,089.4	33.1 %	16.0	70.5	16.5	50.0	
19	1,588.7	79.6 %	49.0	262.1	27.9	30.0	
20	2,457.6	91.0 %	74.8	904.5	46.6	20.0	
21	59.9	0.0 %	10.9	22.2	10.9	50.0	0.0 %
22	1,869.6	31.7 %	15.8	76.4	15.6	50.0	
23	2,567.6	91.4 %	104.4	1,020.1	47.6	10.0	
24	3,020.4	77.5 %	37.8	198.9	19.7	20.0	

En los Cuadros 4.3 y 4.4 se pueden ver los resultados de cada una de las instancias descritas en el Cuadro 4.2: el tiempo de ejecución del programa, el gap y el *makespan* para el modelo lineal y las heurísticas. En el Heurística OA, se presenta el número de ordenamientos aleatorios realizados en las respectivas simulaciones.

Cabe mencionar que, en las instancias 18, 19, 20, 22, 23 y 24, la ejecución del modelo lineal fue detenida y no se llegó a la solución óptima. La heurística Base

solamente llega al óptimo en la instancia 14 y la heurística OA solamente **no** llega al óptimo en las instancias 14 y 15. En las instancias en las que el modelo lineal no llegó al óptimo debido a su tiempo de ejecución, la heurística OA obtuvo mejores resultados en considerablemente menos tiempo que el modelo lineal. Por otro lado, la heurística Base, en mencionadas instancias, solamente obtiene un mejor *makespan* en 3 ocasiones y en ninguna instancia supera los resultados obtenidos por la heurística OA.

Cabe notar que, las heurística Base funciona mejor mientras más trabajos existen, pues es en esas ocasiones es cuando se aproxima al óptimo y supera al modelo lineal limitado por el tiempo de ejecución. Sin embargo, a pesar de tomar más tiempo, aunque no exageradamente mayor, la heurística OA supera a la heurística Base.

A continuación, se presentan tres simulaciones a modo de ejemplo que permitan realizar un análisis comparativo entre las heurísticas y el modelo lineal.

4.2.1. Instancia 10

Para la instancia 10, con 4 máquinas para el primer proceso, 3 para el segundo y 12 trabajos, el modelo tardó 2.56 seg con un makespan de 16.8 unidades de tiempo. Es decir, con el modelo y esas características de la instancia, a la empresa le toma 16.8 horas terminar 12 trabajos.

Para las simulaciones, se trabajaron con variables aleatorias uniformemente distribuidas entre 2 y 10 unidades de tiempo. En el Cuadro 4.5 se pueden ver los tiempos simulados para trabajos y máquinas con los que se trabajaron para la instancia 10.

Cuadro 4.5: Tiempos de realización del trabajo j en la máquina i

	M1				M2		
Trabajo	**A**	**B**	**C**	**D**	**M**	**N**	**O**
1	9.6	9.4	5.6	9.8	-	-	-
2	-	-	-	-	8.3	6.9	2.1
3	6.8	8.8	5.4	7.2	-	-	-
4	-	-	-	-	9.9	9.1	5.8
5	8.1	8.3	6.9	3.5	-	-	-
6	-	-	-	-	3.6	9.6	9.3
7	4.9	2.8	9.7	7.3	-	-	-
8	-	-	-	-	6.1	9.9	6.1
9	5.6	8.6	7.4	8.8	-	-	-
10	-	-	-	-	4.5	7.8	6.4
11	8.9	9.0	2.2	9.8	-	-	-
12	-	-	-	-	7.2	9.3	9.3

En el Cuadro 4.5 cabe especificar, como se mencionó anteriormente, que los trabajos impares corresponden al primer proceso y los trabajos pares al segundo proceso. Se puede ver claramente que las máquinas **A**, **B**, **C** y **D** corresponden al primer proceso, mientras que las máquinas **M**, **N** y **O** al segundo proceso en la *Instancia 10*. En la Figura 4.2 el gráfico de Gantt del modelo lineal con los tiempos de cada uno de los trabajos en el Cuadro 4.6, donde se especifican los tiempos de inicio del trabajo en su respectiva máquina.

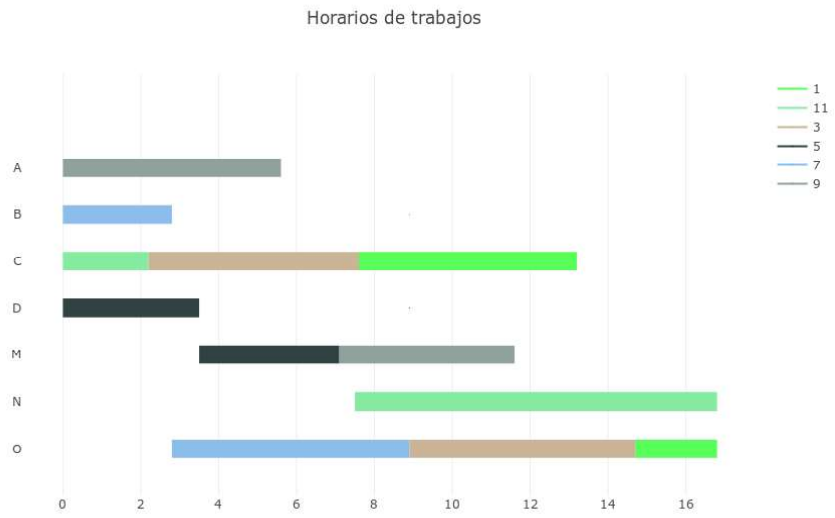

Figura 4.2: Diagrama de Gantt: Modelo, instancia 10

Cuadro 4.6: Tiempos de inicio y fin del trabajo j en la máquina i en el Modelo Lineal para la instancia 10

Máquina	Trabajo	ti	tf
A	9	0.0	5.6
B	7	0.0	2.8
C	11	0.0	2.2
C	1	7.6	13.2
C	3	2.2	7.6
D	5	0.0	3.5
M	5	3.5	7.1
M	9	7.1	11.6
N	11	7.5	16.8
O	7	2.8	8.9
O	1	14.7	16.8
O	3	8.9	14.7

La implementación de las heurísticas tuvieron los siguientes resultados para esta

instancia:

- **Heurística Base:** tomó 0.103 seg en ejecutarse, con un *makespan* de 19.1 horas y el gap de 13.69 %.
- **Heurística OA (50 reordenamientos):** tomó 3.63 seg, con un *makespan* de 16.8 horas y el gap de 0 %.

En las Figuras 4.3 y 4.4 se visualizan los diagramas de Gantt para cada una de las heurísticas con sus respectivos tiempos en los Cuadros 4.7 y 4.8.

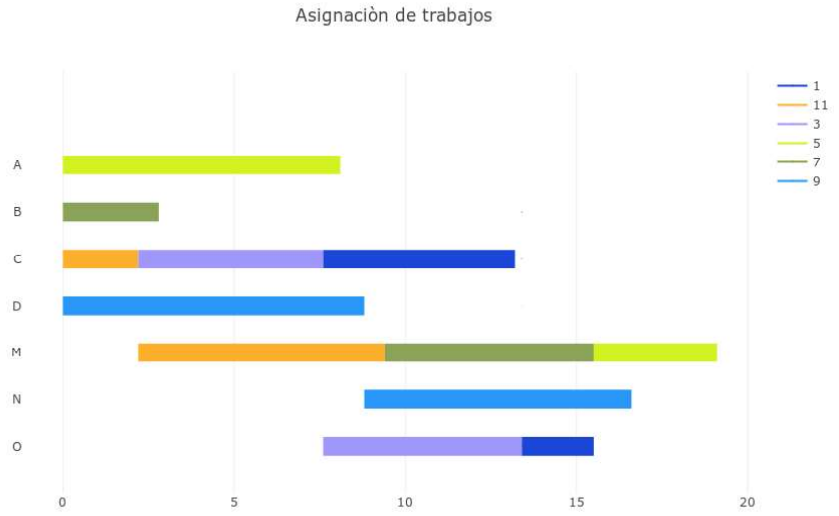

Figura 4.3: Diagrama de Gantt: Heurística Base, instancia 10

Cuadro 4.7: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base para la instancia 10

Máquina	Predecesor	Trabajo	ti	tf
A	0	5	0.0	8.1
B	0	7	0.0	2.8
C	0	11	0.0	2.2
C	11	3	2.2	7.6
C	3	1	7.6	13.2
D	0	9	0.0	8.8
M	0	12	2.2	9.4
M	12	8	9.4	15.5
M	8	6	15.5	19.1
N	0	10	8.8	16.6
O	0	4	7.6	13.4
O	4	2	13.4	15.5

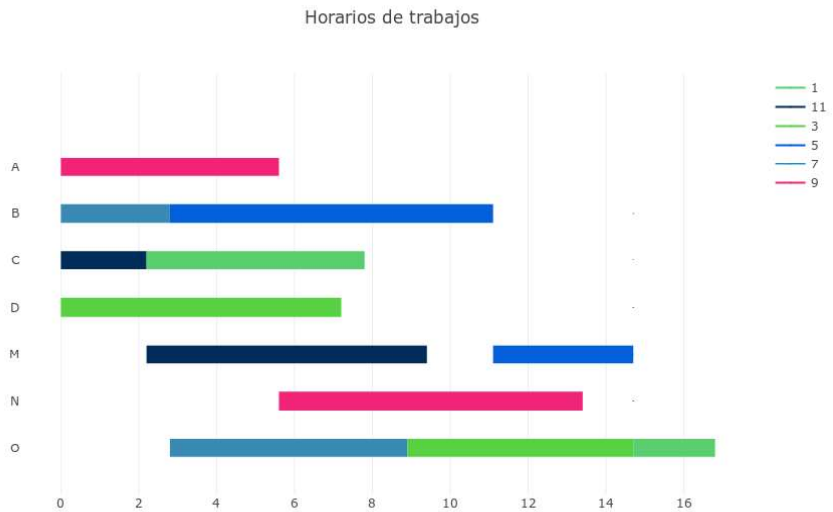

Figura 4.4: Diagrama de Gantt: Heurística OA, instancia 10

Cuadro 4.8: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA para la instancia 10

Máquina	Predecesor	Trabajo	ti	tf
A	0	9	0.0	5.6
B	0	7	0.0	2.8
B	7	5	2.8	11.1
C	0	11	0.0	2.2
C	11	1	2.2	7.8
D	0	3	0.0	7.2
M	0	12	2.2	9.4
M	12	6	11.1	14.7
N	0	10	5.6	13.4
O	0	8	2.8	8.9
O	8	4	8.9	14.7
O	4	2	14.7	16.8

Se puede notar, en esta instancia, cómo, a pesar de tardar más la Heurística OA, esta llega a la solución óptima. En el tiempo 16.8 la instancia alcanza el óptimo al igual que la heurística OA; sin embargo, la heurística en este caso tarda más en llegar al óptimo. Por otro lado, la heurística base en la misma instancia llega a un *makespan* de 19.1 unidades de tiempo en 0.1 segundos.

4.2.2. Instancia 14

Para la instancia 14, con 4 máquinas tanto para el primer proceso como para el segundo y 16 trabajos, el modelo tardó 23.28 seg con un makespan de 13.2 unidades de tiempo. Es decir, con el modelo y esas características de la instancia, a la empresa le toma 13.2 horas terminar 16 trabajos.

En el Cuadro 4.9 se pueden ver los tiempos simulados para trabajos y máquinas con los que se trabajaron para la instancia 14.

Cuadro 4.9: Tiempos de realización del trabajo j en la máquina i

	M1				M2			
Trabajo	**A**	**B**	**C**	**D**	**M**	**N**	**O**	**P**
1	9.6	8.3	7.4	3.9	-	-	-	-
2	-	-	-	-	9.6	6.4	2.5	6.4
3	6.8	2.8	2.2	9.1	-	-	-	-
4	-	-	-	-	9.9	9.3	5.1	4.8
5	8.1	8.6	9.8	2.1	-	-	-	-
6	-	-	-	-	7.8	9.3	5.8	2.9
7	4.9	9.0	7.2	7.2	-	-	-	-
8	-	-	-	-	9.3	7.8	6.1	7.6
9	5.6	5.6	3.5	3.7	-	-	-	-
10	-	-	-	-	2.1	6.2	8.7	8.9
11	8.9	5.4	7.3	2.1	-	-	-	-
12	-	-	-	-	5.8	5.1	9.7	3.3
13	9.4	6.9	8.8	6.2	-	-	-	-
14	-	-	-	-	9.3	3.3	3.8	5.3
15	8.8	9.7	9.8	4.4	-	-	-	-
16	-	-	-	-	6.1	6.1	4.6	3.4

Las máquinas **A**, **B**, **C** y **D** corresponden al primer proceso, mientras que las máquinas **M**, **N**, **O** y **P** al segundo proceso en la *Instancia 14*. En la Figura 4.5 el gráfico de Gantt de los resultados del modelo lineal para la instancia 14 con sus respectivos tiempos por trabajo en el Cuadro 4.10.

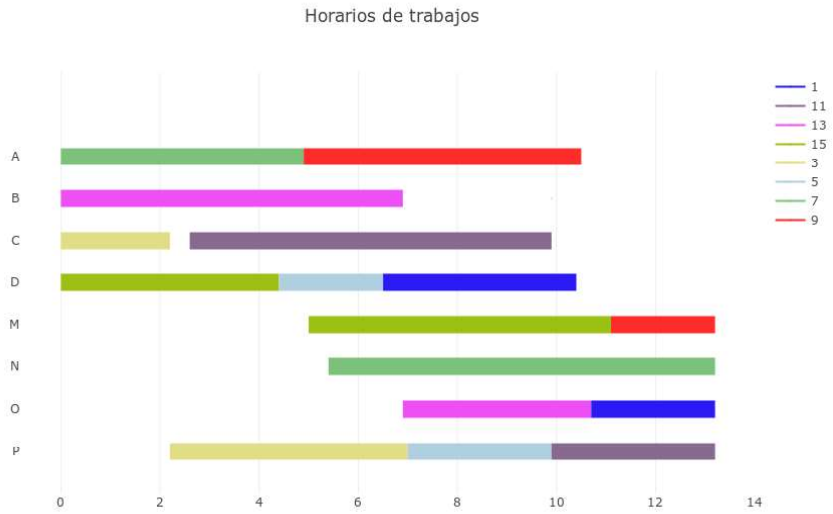

Figura 4.5: Diagrama de Gantt: Modelo, instancia 14

Cuadro 4.10: Tiempos de inicio y fin del trabajo j en la máquina i en el Modelo Lineal para la instancia 14

Máquina	Trabajo	ti	tf
A	7	0.0	4.9
A	9	4.9	10.5
B	13	0.0	6.9
C	3	0.0	2.2
C	11	2.6	9.9
D	15	0.0	4.4
D	1	6.5	10.4
D	5	4.4	6.5
M	15	5.0	11.1
M	9	11.1	13.2
N	7	5.4	13.2
O	13	6.9	10.7
O	1	10.7	13.2
P	3	2.2	7.0
P	5	7.0	9.9
P	11	9.9	13.2

La implementación de las heurísticas tuvieron los siguientes resultados para esta instancia:

- **Heurística Base:** tomó 0.2 seg en ejecutarse, con un *makespan* de 15.3 horas y el gap de 15.91 %.
- **Heurística OA (50 reordenamientos):** tomó 9.28 seg, con un *makespan* de 13.5 horas y el gap de 2.27 %.

En las Figuras 4.6 y 4.7 se visualizan los diagramas de Gantt para cada una de las heurísticas con los tiempos de inicio y finalización de los trabajos en sus máquinas asignadas en los Cuadros 4.11 y 4.12.

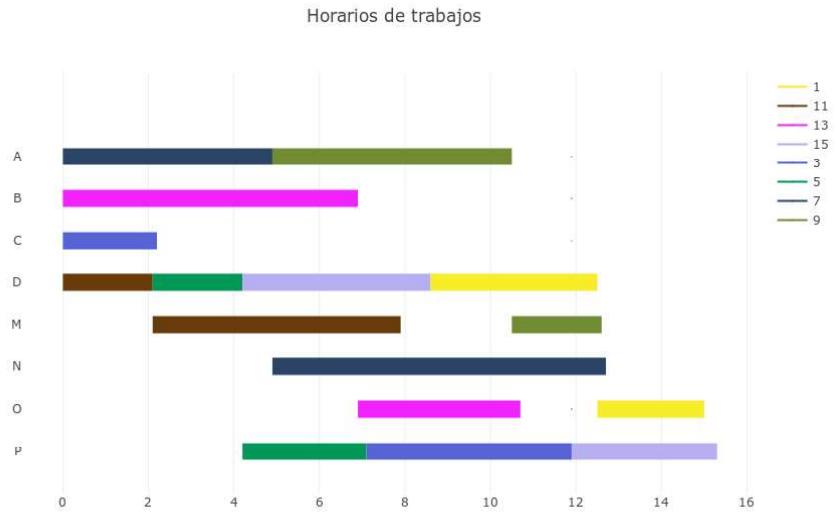

Figura 4.6: Diagrama de Gantt: Heurística Base, instancia 14

Cuadro 4.11: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base para la instancia 14

Máquina	Predecesor	Trabajo	ti	tf
A	0	7	0.0	4.9
A	7	9	4.9	10.5
B	0	13	0.0	6.9
C	0	3	0.0	2.2
C	11	5	2.1	4.2
C	5	15	4.2	8.6
C	15	1	8.6	12.5
D	0	11	0.0	2.1
M	0	12	2.1	7.9
M	12	10	10.5	12.6
N	0	8	4.9	12.7
O	0	14	6.9	10.7
O	14	2	12.5	15.0
P	0	6	4.2	7.1
P	6	4	7.1	11.9
P	4	16	11.9	15.3

Figura 4.7: Diagrama de Gantt: Heurística OA, instancia 14

Cuadro 4.12: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA para la instancia 14

Máquina	Predecesor	Trabajo	ti	tf
A	0	7	0.0	4.9
B	0	3	0.0	2.8
B	3	13	2.8	9.7
C	0	9	0.0	3.5
C	9	1	3.5	10.9
D	0	11	0.0	2.1
D	11	15	2.1	6.5
D	15	5	6.5	8.6
M	0	10	3.5	5.6
M	10	16	6.5	12.6
N	0	12	2.1	7.2
N	12	14	9.7	13.0
O	0	8	4.9	11.0
O	8	2	11.0	13.5
P	0	4	2.8	7.6
P	4	6	8.6	11.5

En esta instancia ninguna de las heurísticas llega al óptimo de 13.2. Sin embargo, el gap de 15.91 % de la heurística Base no es comparable con el 2.27 % de la heurística OA. Se evidencia la mejor aproximación que tiene la heurística OA con relación a la heurística Base al modelo lineal.

4.2.3. Instancia 19

Para la instancia 19, con 5 máquinas para el primer proceso, 4 para el segundo y 50 trabajos, el modelo fue detenido en 1588.7 seg con un makespan de 49 unidades de tiempo y un gap de 79.6 %. Los resultados son parciales pues en este caso programa fue detenido por su tiempo de ejecución, demostrando su no factibilidad

práctica.

En los Cuadros 4.13 y 4.13 se pueden ver los tiempos simulados para trabajos y máquinas con los que se trabajaron para la instancia 19.

Cuadro 4.13: Tiempos de realización del trabajo j en la máquina i

	M1					M2			
Trabajo	**A**	**B**	**C**	**D**	**E**	**M**	**N**	**O**	**P**
1	9.6	9.1	2.4	6.6	4.0	-	-	-	-
2	-	-	-	-	-	2.4	8.4	7.2	9.7
3	6.8	2.1	8.6	2.8	5.2	-	-	-	-
4	-	-	-	-	-	6.6	4.7	7.8	8.0
5	8.1	7.2	2.2	2.5	4.4	-	-	-	-
6	-	-	-	-	-	7.1	7.8	4.4	2.8
7	4.9	3.7	3.2	4.6	3.4	-	-	-	-
8	-	-	-	-	-	3.5	6.4	2.7	9.6
9	5.6	2.1	2.2	3.9	7.8	-	-	-	-
10	-	-	-	-	-	9.4	3.4	4.7	7.5
11	8.9	6.2	9.4	4.4	3.1	-	-	-	-
12	-	-	-	-	-	6.3	7.1	7.9	6.4
13	9.4	4.4	10.0	3.7	6.3	-	-	-	-
14	-	-	-	-	-	4.2	3.0	3.7	6.0
15	8.8	7.8	7.0	8.3	2.1	-	-	-	-
16	-	-	-	-	-	2.6	6.9	9.9	8.1
17	8.3	5.2	9.1	9.3	4.6	-	-	-	-
18	-	-	-	-	-	5.6	6.4	4.0	6.7
19	2.8	5.6	3.3	9.0	8.7	-	-	-	-
20	-	-	-	-	-	6.7	8.9	4.2	9.6
21	8.6	4.8	9.5	7.8	7.1	-	-	-	-
22	-	-	-	-	-	2.1	7.7	4.3	9.2
23	9.0	2.5	9.0	6.3	2.2	-	-	-	-
24	-	-	-	-	-	6.5	5.9	5.6	5.9
25	5.6	5.8	2.8	8.8	9.4	-	-	-	-

Cuadro 4.14: Tiempos de realización del trabajo j en la máquina i

	M1					M2			
Trabajo	**A**	**B**	**C**	**D**	**E**	**M**	**N**	**O**	**P**
31	9.7	6.3	6.8	7.5	2.9	-	-	-	-
26	-	-	-	-	-	6.4	3.1	4.2	3.0
27	5.4	2.3	2.0	2.8	4.5	-	-	-	-
28	-	-	-	-	-	7.0	2.0	3.1	3.6
29	6.9	9.4	5.9	2.6	7.3	-	-	-	-
30	-	-	-	-	-	10.0	3.6	5.2	4.1
32	-	-	-	-	-	4.2	4.1	4.5	6.1
33	7.4	8.8	9.7	3.1	8.6	-	-	-	-
34	-	-	-	-	-	6.9	6.8	4.2	2.8
35	2.2	4.0	5.6	4.8	8.0	-	-	-	-
36	-	-	-	-	-	7.6	8.8	4.6	6.9
37	9.8	7.5	3.3	5.8	7.8	-	-	-	-
38	-	-	-	-	-	10.0	8.8	7.0	6.7
39	7.2	4.4	4.1	5.8	4.4	-	-	-	-
40	-	-	-	-	-	9.2	8.1	5.0	3.6
41	3.5	3.4	9.2	5.3	6.0	-	-	-	-
42	-	-	-	-	-	7.9	9.5	9.7	2.5
43	7.3	2.0	9.5	2.2	4.1	-	-	-	-
44	-	-	-	-	-	6.1	3.6	8.7	4.0
45	8.8	8.1	9.5	9.4	2.7	-	-	-	-
46	-	-	-	-	-	7.9	5.8	9.4	3.1
47	9.8	7.4	8.0	5.6	5.4	-	-	-	-
48	-	-	-	-	-	7.1	5.7	7.3	4.1
49	3.9	9.0	7.2	4.4	5.3	-	-	-	-
50	-	-	-	-	-	8.0	4.0	3.4	6.6

Las máquinas **A**, **B**, **C**, **D** y **E** corresponden al primer proceso, mientras que las máquinas **M**, **N**, **O** y **P** al segundo proceso en la *Instancia 14*. En la Figura 4.8 el gráfico de Gantt de los resultados del modelo lineal para la instancia 19 con sus respectivos tiempos por trabajo en el Cuadro 4.15.

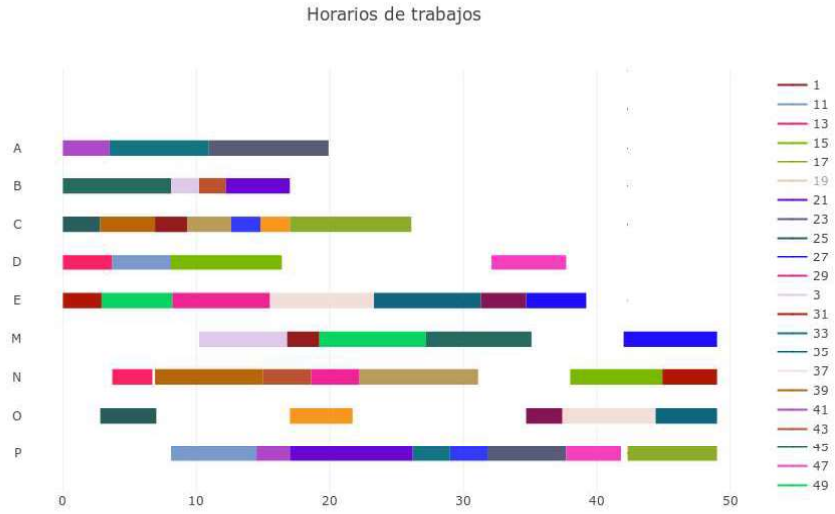

Figura 4.8: Diagrama de Gantt: Modelo, instancia 19

Cuadro 4.15: Tiempos de inicio y fin del trabajo j en la máquina i en el Modelo Lineal para la instancia 19

Máquina	Trabajo	ti	tf	Máquina	Trabajo	ti	tf
A	41	0.0	3.5	M	3	10.2	16.8
A	23	10.9	19.9	M	49	19.2	27.2
A	33	3.5	10.9	M	1	16.8	19.2
B	45	0.0	8.1	M	27	42.0	49.0
B	43	10.2	12.2	M	45	27.2	35.1
B	21	12.2	17.0	N	13	3.7	6.7
B	3	8.1	10.2	N	39	6.9	15.0
C	25	0.0	2.8	N	31	44.9	49.0
C	19	9.3	12.6	N	15	38.0	44.9
C	9	14.8	17.0	N	19	22.2	31.1
C	17	17.0	26.1	N	43	15.0	18.6
C	5	12.6	14.8	N	29	18.6	22.2
C	39	2.8	6.9	O	25	2.8	7.0
C	1	6.9	9.3	O	37	37.4	44.4
D	13	0.0	3.7	O	7	34.7	37.4
D	15	8.1	16.4	O	9	17.0	21.7
D	11	3.7	8.1	O	35	44.4	49.0
D	47	32.1	37.7	P	11	8.1	14.5
E	31	0.0	2.9	P	23	31.8	37.7
E	27	34.7	39.2	P	41	14.5	17.0
E	37	15.5	23.3	P	33	26.2	29.0
E	49	2.9	8.2	P	47	37.7	41.8
E	7	31.3	34.7	P	5	29.0	31.8
E	35	23.3	31.3	P	21	17.0	26.2
E	29	8.2	15.5	P	17	42.3	49.0

La implementación de las heurísticas tuvieron los siguientes resultados para esta instancia:

- **Heurística Base:** tomó 8.19 seg en ejecutarse, con un *makespan* de 32.8 horas.
- **Heurística OA (30 reordenamientos):** tomó 262.08 seg, con un *makespan* de 27.9 horas.

En las Figuras 4.9 y 4.10 se visualizan los diagramas de Gantt para cada una de las heurísticas con los tiempos de inicio y finalización de los trabajos en sus máquinas asignadas en los Cuadros 4.16, 4.17 para la heurística Base y en los Cuadros 4.18 y 4.19 para la heurística OA.

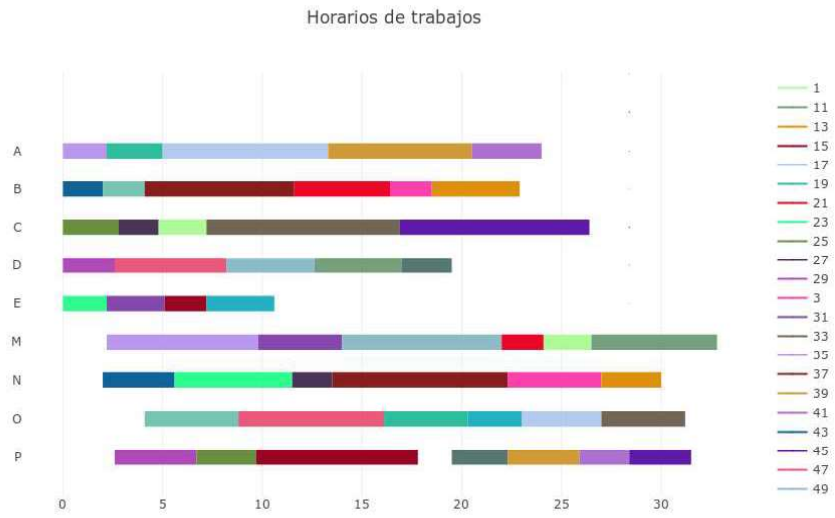

Figura 4.9: Diagrama de Gantt: Heurística Base, instancia 19

Cuadro 4.16: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base para la instancia 19

Máquina	Predecesor	Trabajo	ti	tf
A	0	35	0.0	2.2
A	35	19	2.2	5.0
A	19	17	5.0	13.3
A	17	39	13.3	20.5
A	39	41	20.5	24.0
B	0	43	0.0	2.0
B	43	9	2.0	4.1
B	9	37	4.1	11.6
B	37	21	11.6	16.4
B	21	3	16.4	18.5
B	3	13	18.5	22.9
C	0	25	0.0	2.8
C	25	27	2.8	4.8
C	27	1	4.8	7.2
C	1	33	7.2	16.9
C	33	45	16.9	26.4
D	0	29	0.0	2.6
D	29	47	2.6	8.2
D	47	49	8.2	12.6
D	49	11	12.6	17.0
D	11	5	17.0	19.5
E	0	23	0.0	2.2
E	23	31	2.2	5.1
E	31	15	5.1	7.2

Cuadro 4.17: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base para la instancia 19

Máquina	Predecesor	Trabajo	ti	tf
E	15	7	7.2	10.6
M	0	36	2.2	9.8
M	36	32	9.8	14.0
M	32	50	14.0	22.0
M	50	22	22.0	24.1
M	22	2	24.1	26.5
M	2	12	26.5	32.8
N	0	44	2.0	5.6
N	44	24	5.6	11.5
N	24	28	11.5	13.5
N	28	38	13.5	22.3
N	38	4	22.3	27.0
N	4	14	27.0	30.0
O	0	10	4.1	8.8
O	10	48	8.8	16.1
O	48	20	16.1	20.3
O	20	8	20.3	23.0
O	8	18	23.0	27.0
O	18	34	27.0	31.2
P	0	30	2.6	6.7
P	30	26	6.7	9.7
P	26	16	9.7	17.8
P	16	6	19.5	22.3
P	6	40	22.3	25.9
P	40	42	25.9	28.4
P	42	46	28.4	31.5

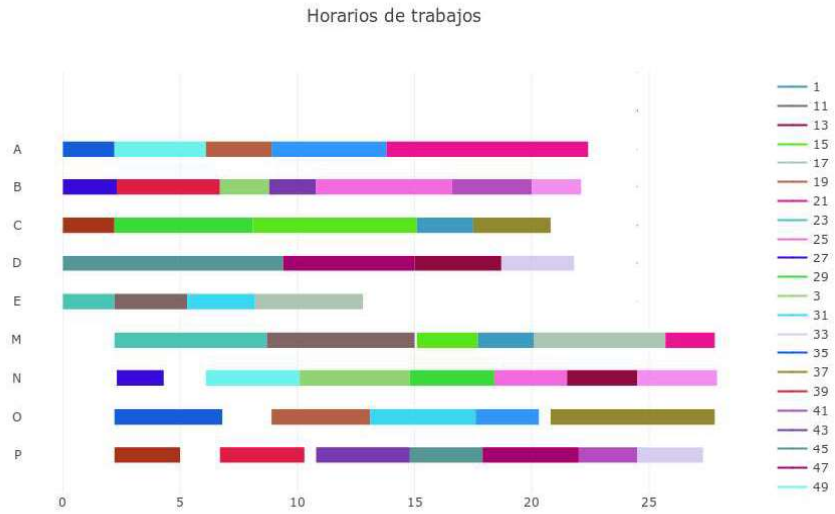

Figura 4.10: Diagrama de Gantt: Heurística OA, instancia 19

Cuadro 4.18: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA para la instancia 19

Máquina	Predecesor	Trabajo	ti	tf
A	0	35	0.0	2.2
A	35	49	2.2	6.1
A	49	19	6.1	8.9
A	19	7	8.9	13.8
A	7	21	13.8	22.4
B	0	27	0.0	2.3
B	27	39	2.3	6.7
B	39	3	6.7	8.8
B	3	43	8.8	10.8
B	43	25	10.8	16.6
B	25	41	16.6	20.0
B	41	9	20.0	22.1
C	0	5	0.0	2.2
C	5	29	2.2	8.1
C	29	15	8.1	15.1
C	15	1	15.1	17.5
C	1	37	17.5	20.8
D	0	45	0.0	9.4
D	45	47	9.4	15.0
D	47	13	15.0	18.7
D	13	33	18.7	21.8
E	0	23	0.0	2.2
E	23	11	2.2	5.3
E	11	31	5.3	8.2
E	31	17	8.2	12.8

Cuadro 4.19: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA para la instancia 19

Máquina	Predecesor	Trabajo	ti	tf
M	0	24	2.2	8.7
M	24	12	8.7	15.0
M	12	16	15.1	17.7
M	16	2	17.7	20.1
M	2	18	20.1	25.7
M	18	22	25.7	27.8
N	0	28	2.3	4.3
N	28	50	6.1	10.1
N	50	4	10.1	14.8
N	4	30	14.8	18.4
N	30	26	18.4	21.5
N	26	14	21.5	24.5
N	14	10	24.5	27.9
O	0	36	2.2	6.8
O	36	20	8.9	13.1
O	20	32	13.1	17.6
O	32	8	17.6	20.3
O	8	38	20.8	27.8
P	0	6	2.2	5.0
P	6	40	6.7	10.3
P	40	44	10.8	14.8
P	44	46	14.8	17.9
P	46	48	17.9	22.0
P	48	42	22.0	24.5
P	42	34	24.5	27.3

Al ser detenido el modelo lineal por exceso de tiempo de ejecución, se desconoce el óptimo. Sin embargo, en los más de 20 minutos de ejecución, el modelo lineal alcanza un resultado aún peor que el hallado por la heurística Base. Además, la diferencia de 5 horas entre el *makespan* de la heurística Base y el *makespan* de la heurística OA es determinando en la elección del mejor resultado a pesar de que la heurística OA tarde cerca de 8 minutos en su ejecución.

Como nota final a esta sección, cabe mencionar que, de las 24 simulaciones realizadas, solamente en 1 caso la heurística Base logra llegar al resultado óptimo. Eso sí, salvo en las instancias más grandes, no toma más de 1 minuto de encontrar una solución. El modelo, por otro lado, en cuanto supera los 16 trabajos, su tiempo de procesamiento incrementa, llegando a tardar 40 minutos en 100 trabajos. Por otro lado, la Heurística OA, aunque tarda más que la heurística Base en las 24 simulaciones, alcanza el óptimo en 15 ocasiones con un promedio de 5 segundos para los casos "pequeños".

4.3. Resultados

Los resultados obtenidos con el modelo y las heurísticas implementadas, se basaron en la información de ventas semanal de la empresa. Es decir, el volumen producido considera solamente valores de una semana arbitrariamente tomada.

A lo largo de esa semana se tuvo el volumen de pedidos mostrando en el Cuadro 4.20. Cada uno de los volúmenes del Cuadro 4.20 reflejan un trabajo en el modelo lineal y su tiempo total de ejecución en el programa equivale al tiempo unitario de ese tipo de producto por el volumen. Esto debido a que cada máquina procesa un solo producto a la vez. Así, en la columna Pedido se especifica el volumen del pedido y en la columna Trabajo se especifica los índices de trabajos realizados en las máquinas que se verá luego en el gráfico de Gantt.

Cuadro 4.20: Volumen de pedidos de la empresa

Producto	Segmento de producto	Pedido	Trabajo
Gorras	Menos de 3000 puntadas	30	1-2
Gorras	Más de 3000 puntadas	25	3-4
XS	Menos de 3000 puntadas	15	5-6
XS	Entre 12000 y 15000 puntadas	30	7-8
XS	Más de 15000 puntadas	15	9-10
S	Entre 9000 y 12000 puntadas	30	11-12
S	Entre 9000 y 12000 puntadas	25	13-14
S	Entre 12000 y 15000 puntadas	35	15-16
M	Entre 3000 y 6000 puntadas	40	17-18
M	Entre 3000 y 6000 puntadas	35	19-20
M	Entre 6000 y 9000 puntadas	20	21-22
M	Entre 6000 y 9000 puntadas	30	23-24
M	Entre 6000 y 9000 puntadas	35	25-26
M	Entre 12000 y 15000 puntadas	40	27-28
L	Entre 6000 y 9000 puntadas	35	29-30
L	Entre 9000 y 12000 puntadas	45	31-32
L	Entre 9000 y 12000 puntadas	20	33-34
XL	Menos de 3000 puntadas	30	35-36
XL	Entre 6000 y 9000 puntadas	35	37-38
XL	Más de 15000 puntadas	25	39-40

En los Cuadros 4.21 y 4.22, se verifican los tiempos de cada uno de los trabajos especificados anteriormente para cada una de las máquinas.

Cuadro 4.21: Tiempos de realización del trabajo j en la máquina i

Trabajo	A	B	C	M	N	O
1	203.9	195.2	207.6	-	-	-
2	-	-	-	145.7	175.4	180.7
3	169.9	162.7	173.0	-	-	-
4	-	-	-	196.9	230.1	233.2
5	98.3	97.5	99.0	-	-	-
6	-	-	-	46.7	63.4	64.6
7	196.6	194.9	197.9	-	-	-
8	-	-	-	458.9	490.0	495.4
9	98.3	97.5	99.0	-	-	-
10	-	-	-	287.3	304.2	306.0
11	207.2	209.3	206.5	-	-	-
12	-	-	-	301.6	332.5	336.6
13	172.7	174.4	172.0	-	-	-
14	-	-	-	251.3	277.1	280.5
15	241.7	244.2	240.9	-	-	-
16	-	-	-	535.4	571.7	577.9
17	291.9	293.4	292.4	-	-	-
18	-	-	-	195.5	243.8	235.1
19	255.4	256.8	255.9	-	-	-
20	-	-	-	171.1	213.3	205.7
21	146.0	146.7	146.2	-	-	-
22	-	-	-	140.1	160.3	165.1
23	218.9	220.1	219.3	-	-	-
24	-	-	-	210.2	240.4	247.7
25	255.4	256.8	255.9	-	-	-
26	-	-	-	245.2	280.5	289.0

Cuadro 4.22: Tiempos de realización del trabajo j en la máquina i

Trabajo	A	B	C	M	N	O
27	291.9	293.4	292.4	-	-	-
28	-	-	-	611.9	653.4	660.5
29	266.9	272.5	265.3	-	-	-
30	-	-	-	245.2	280.5	289.0
31	343.2	350.3	341.1	-	-	-
32	-	-	-	315.3	360.7	371.6
33	152.5	155.7	151.6	-	-	-
34	-	-	-	201.1	221.7	224.4
35	250.3	245.5	248.3	-	-	-
36	-	-	-	93.4	126.7	129.1
37	292.0	286.4	289.7	-	-	-
38	-	-	-	245.2	280.5	289.0
39	208.6	204.6	206.9	-	-	-
40	-	-	-	478.8	507.1	510.1

En el Cuadro 4.23 se pueden ver los resultados comparativos de los tiempos de ejecución, el GAP y el *makespan* del modelo y las heurísticas implementadas.

Cuadro 4.23: Tiempo de ejecución, GAP y Makespan del modelo y las heurísticas

	Tiempo de ejecución (s)	**GAP (%)**	**Makespan (s)**
Modelo Lineal	1798.33	60.3	2277.2
Heurística Base	3.65		2321.9
Heurística OA	103.34		2140.6

Luego de aproximadamente 20 minutos el modelo lineal paró su ejecución, llegando a un *makespan* de 2277.2 segundos. Por otro lado, las heurísticas Base y OA se ejecutaron en 3.65 y 103.34 segundos (debido a sus 50 repeticiones), respectivamente. La heurística OA tarda un poco más que la heurística Base, sin embargo el

makespan es claramente mejor tanto que de la heurística Base como del modelo lineal (en el tiempo que corrió el algoritmo). Por otro lado, el margen que existe entre el la heurística Base y el modelo lineal, en su limitado tiempo de ejecución, es de 44.7 segundos en el *makespan* a favor del modelo lineal.

Con estos resultados se puede ver claramente que la heurística OA llega, en un tiempo prudencial, a resultados factibles en comparación con el modelo lineal.

Es importante recalcar que, a pesar de ser la heurística Base la más rápida en ejecutarse, por los resultados obtenidos en las simulaciones y en los resultados, no puede elegirse para alcanzar el óptimo ni aun en casos pequeños como se vio en la revisión de la instancia 14.

En las Figuras 4.11, 4.12 y 4.13 y sus respectivos tiempos en los Cuadros 4.24, 4.25, 4.26, 4.27 y 4.28 se pueden ver los gráficos de Gantt para cada uno de los algoritmos implementados.

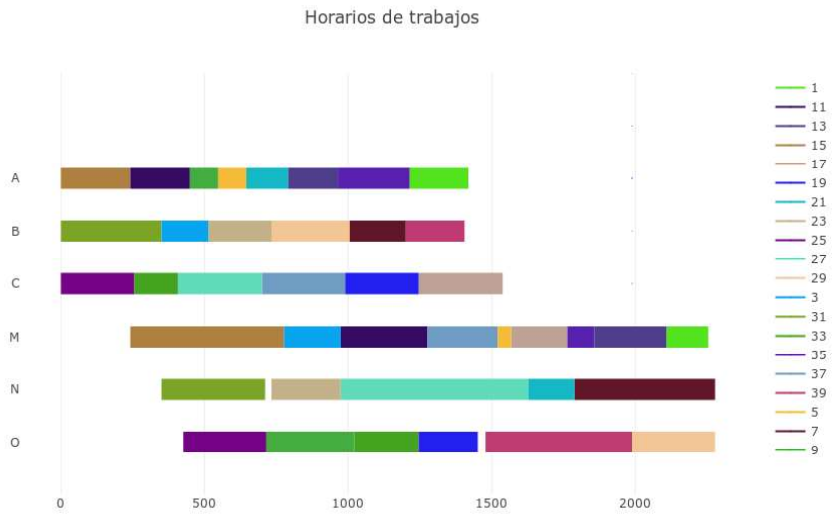

Figura 4.11: Diagrama de Gantt: Modelo Lineal

Cuadro 4.24: Tiempos de inicio y fin del trabajo j en la máquina i en el modelo lineal

Máquina	Trabajo	ti	tf	Máquina	Trabajo	ti	tf
A	15	-	241.7	**M**	15	241.7	777.1
A	21	645.5	791.5	**M**	11	974.0	1,275.6
A	5	547.2	645.5	**M**	17	1,567.5	1,763.0
A	9	448.9	547.2	**M**	37	1,275.6	1,520.8
A	35	964.2	1,214.5	**M**	1	2,107.7	2,253.4
A	11	241.7	448.9	**M**	3	777.1	974.0
A	13	791.5	964.2	**M**	35	1,763.0	1,856.4
A	1	1,214.5	1,418.4	**M**	13	1,856.4	2,107.7
B	31	-	350.3	**M**	5	1,520.8	1,567.5
B	23	513.0	733.1	**N**	31	350.3	711.0
B	39	1,200.5	1,405.1	**N**	7	1,787.2	2,277.2
B	29	733.1	1,005.6	**N**	27	973.5	1,626.9
B	7	1,005.6	1,200.5	**N**	21	1,626.9	1,787.2
B	3	350.3	513.0	**N**	23	733.1	973.5
C	25	-	255.9	**O**	25	426.1	715.1
C	17	1,245.5	1,537.9	**O**	33	1,021.1	1,245.5
C	33	255.9	407.5	**O**	39	1,478.1	1,988.2
C	37	699.9	989.6	**O**	9	715.1	1,021.1
C	27	407.5	699.9	**O**	19	1,245.5	1,451.2
C	19	989.6	1,245.5	**O**	29	1,988.2	2,277.2

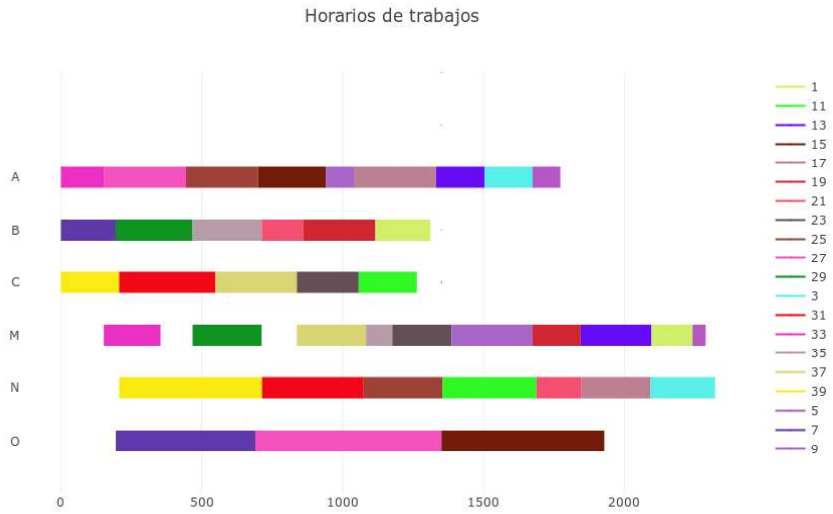

Figura 4.12: Diagrama de Gantt: Heurística Base

Cuadro 4.25: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base

Máquina	Predecesor	Trabajo	ti	tf
A	0	33	0	152.5
A	33	27	152.5	444.4
A	27	25	444.4	699.8
A	25	15	699.8	941.5
A	15	9	941.5	1039.8
A	9	17	1039.8	1331.7
A	17	13	1331.7	1504.4
A	13	3	1504.4	1674.3
A	3	5	1674.3	1772.6
B	0	7	0	194.9
B	7	29	194.9	467.4
B	29	35	467.4	712.9
B	35	21	712.9	859.6
B	21	19	859.6	1116.4

Cuadro 4.26: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística Base

Máquina	Predecesor	Trabajo	ti	tf
B	19	1	1116.4	1311.6
C	0	39	0	206.9
C	39	31	206.9	548
C	31	37	548	837.7
C	37	23	837.7	1057
C	23	11	1057	1263.5
M	0	34	152.5	353.6
M	34	30	467.4	712.6
M	30	38	837.7	1082.9
M	38	36	1082.9	1176.3
M	36	24	1176.3	1386.5
M	24	10	1386.5	1673.8
M	10	20	1673.8	1844.9
M	20	14	1844.9	2096.2
M	14	2	2096.2	2241.9
M	2	6	2241.9	2288.6
N	0	40	206.9	714
N	40	32	714	1074.7
N	32	26	1074.7	1355.2
N	26	12	1355.2	1687.7
N	12	22	1687.7	1848
N	22	18	1848	2091.8
N	18	4	2091.8	2321.9
O	0	8	194.9	690.3
O	8	28	690.3	1350.8
O	28	16	1350.8	1928.7

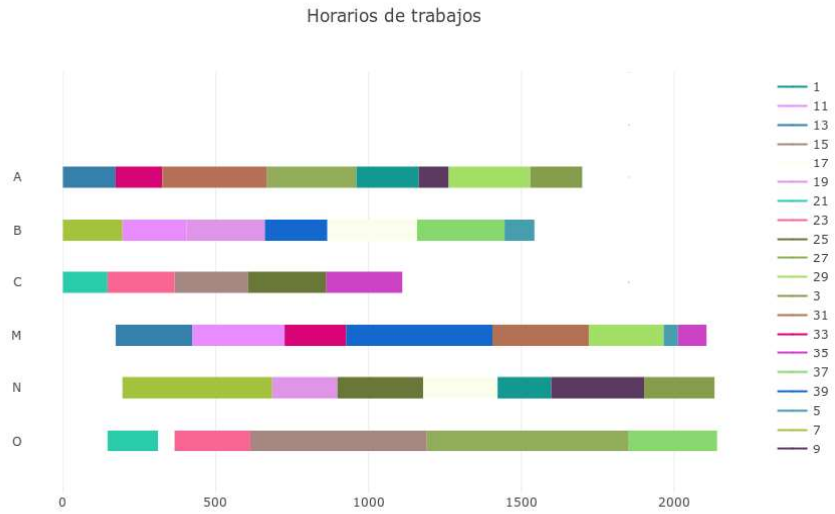

Figura 4.13: Diagrama de Gantt: Heurística OA

Cuadro 4.27: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA

Máquina	Predecesor	Trabajo	ti	tf
A	0	13	0	172.7
A	13	33	172.7	325.2
A	33	31	325.2	668.4
A	31	27	668.4	960.3
A	27	1	960.3	1164.2
A	1	9	1164.2	1262.5
A	9	29	1262.5	1529.4
A	29	3	1529.4	1699.3
B	0	7	0	194.9
B	7	11	194.9	404.2
B	11	19	404.2	661
B	19	39	661	865.6
B	39	17	865.6	1159

Cuadro 4.28: Tiempos de inicio y fin del trabajo j en la máquina i en la heurística OA

Máquina	Predecesor	Trabajo	ti	tf
B	17	37	1159	1445.4
B	37	5	1445.4	1542.9
C	0	21	0	146.2
C	21	23	146.2	365.5
C	23	15	365.5	606.4
C	15	25	606.4	862.3
C	25	35	862.3	1110.6
M	0	14	172.7	424
M	14	12	424	725.6
M	12	34	725.6	926.7
M	34	40	926.7	1405.5
M	40	32	1405.5	1720.8
M	32	30	1720.8	1966
M	30	6	1966	2012.7
M	6	36	2012.7	2106.1
N	0	8	194.9	684.9
N	8	20	684.9	898.2
N	20	26	898.2	1178.7
N	26	18	1178.7	1422.5
N	18	2	1422.5	1597.9
N	2	10	1597.9	1902.1
N	10	4	1902.1	2132.2
O	0	22	146.2	311.3
O	22	24	365.5	613.2
O	24	16	613.2	1191.1
O	16	28	1191.1	1851.6
O	28	38	1851.6	2140.6

Para los resultados, se hicieron cálculos con base en una semana entera de trabajo. Como se mencionó, de acuerdo a la selección aleatoria de una semana de trabajo se fijó la distribución. Se puede ver en los gráficos y en el Cuadro 4.23 que la Heurística OA llega a un resultado satisfactorio con un *makespan* considerablemente menor a la Heurística Base y el Modelo Lineal. Ciertamente, el Modelo Lineal no llega al óptimo al parar su ejecución luego de 20 minutos; sin embargo, hay que considerar que, en la práctica, es poco factible la espera de más de 10 minutos por la organización de eventos que se deben recalcular cada vez que llega un pedido o un conjunto de pedidos. Resultaría factible en caso de que se plantee un reordenamiento de la asignación al final del día, dejando que el programa se ejecute durante las horas no laborables. Aunque ese planteamiento no sirve en caso de determinar fechas de entrega sobre la marcha a los clientes. Por ello, es importante poder encontrar una estimación del tiempo de entrega al momento de llegar el pedido y resulta preferible hacer esperar al cliente alrededor de 3 minutos por darle una estimación certera de la fecha de entrega.

Siendo el objetivo la estimación de los tiempos de entrega, cabe mencionar la factibilidad de la heurística OA tanto en tiempo de ejecución equivalente al tiempo de permanencia como su aproximación a la solución óptima. Por ello, se implementó una nueva restricción a la heurística OA para poder estimar los tiempos de entrega. La restricción involucra fijar los tiempos de finalización de los trabajos pendientes de modo que, al ingresar nuevos trabajos en el tiempo t, en la reorganización del horario de trabajo de las máquinas, los trabajos pendientes no sean afectados. Así, se ingresaron 3 nuevos trabajos en los resultados obtenidos previamente y el nuevo horario con sus respectivos tiempos pueden verse en la Figura 4.14 y el Cuadro 4.29.

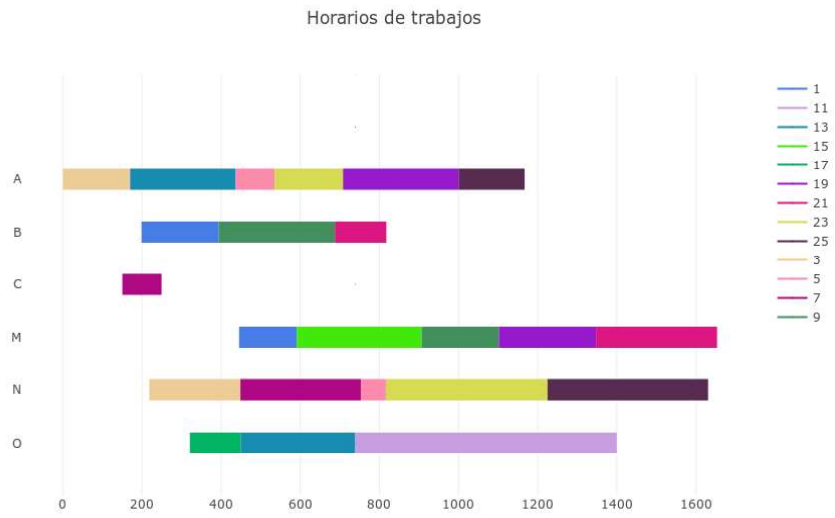

Figura 4.14: Diagrama de Gantt: Reordenamiento de los horarios con nuevos trabajos

Cuadro 4.29: Tiempos de inicio y fin del trabajo j en la máquina i para los trabajos pendientes y los trabajos ingresados en un tiempo t

Máquina	**Predecesor**	**Trabajo**	**ti**	**tf**
A	0	17	0.3	0.3
A	17	3	0.3	170.2
A	3	13	170.2	437.1
A	13	5	437.1	535.4
A	5	15	535.4	535.4
A	15	23	535.4	708.1
A	23	11	708.1	708.1
A	11	19	708.1	1000.1
A	19	25	1000.1	1166.9
B	0	1	199	394.2
B	1	9	394.2	687.6
B	9	21	687.6	817.5
C	0	7	150.6	249.6
M	0	2	445.5	591.2
M	2	16	591.2	906.5
M	16	10	906.5	1102
M	10	20	1102	1347.2
M	20	22	1347.2	1653.1
N	0	4	218.7	448.8
N	4	8	448.8	753
N	8	6	753	816.4
N	6	24	816.4	1224.8
N	24	26	1224.8	1630.4
O	0	18	321.1	450.2
O	18	14	450.2	739.2
O	14	12	739.2	1399.7

Los tiempos que toma realizar los nuevos trabajos por máquina en el modelo se pueden verificar en el Cuadro 4.30.

Cuadro 4.30: Tiempos de realización de los nuevos trabajo j en la máquina i

Trabajo	A	B	C	M	N	O
1	131.1	129.9	132.0	0.0	0.0	0.0
2	0.0	0.0	0.0	305.9	326.7	330.2
3	172.7	174.4	172.0	0.0	0.0	0.0
4	0.0	0.0	0.0	382.4	408.4	412.8
5	166.8	163.7	165.6	0.0	0.0	0.0
6	0.0	0.0	0.0	383.1	405.6	408.1

Se supone que los nuevos trabajos son ingresados después de 2 días en el horario de trabajos. Como se mencionó anteriormente, los trabajos pendientes mantienen su *due date* de modo que el ingreso de los nuevos productos no afecten esa fecha de entrega.

Además, debido a que en el segundo día existen trabajos pendientes por finalizar, se debe ingresar un tiempo inicial de procesamiento por máquina, es decir, el primer trabajo comenzará después de ese tiempo previo.

Par ello se debe calcular los tiempos iniciales de acuerdo a los trabajos pendientes por máquina hasta ese momento:

- Máquina A: se terminaron los trabajos 13, 31 y 33. El trabajo 27 quedó con un tiempo de 0.3 que se considera como tiempo inicial al ingreso de los nuevos trabajos.
- Máquina B: se terminaron los trabajos 7, 11, 19 y 39. El trabajo 17 quedó con un tiempo de 199 que se considera como tiempo inicial al ingreso de los nuevos trabajos.
- Máquina C: se terminaron los trabajos 15, 21, 23 y 25. El trabajo 35 quedó con un tiempo de 150.6 que se considera como tiempo inicial al ingreso de los nuevos trabajos.
- Máquina M: se terminaron los trabajos 12, 14, 34. El trabajo 40 quedó con un tiempo de 218.7 que se considera como tiempo inicial al ingreso de los nuevos

trabajos.

- Máquina N: se terminaron los trabajos 8 y 20. El trabajo 26 quedó con un tiempo de 218.7 que se considera como tiempo inicial al ingreso de los nuevos trabajos.
- Máquina O: se terminaron los trabajos 22 y 24. El trabajo 16 quedó con un tiempo de 231.1 que se considera como tiempo inicial al ingreso de los nuevos trabajos.

Entonces, para los nuevos trabajos, se estiman los siguientes tiempos de entrega, considerando la carga de trabajos pendiente:

- Para el trabajo 1 (21 y 22), se estima que se termine en 1653.1 minutos, es decir, en 3 días y medio aproximadamente.
- Para el trabajo 2 (23 y 24), se estima que se entregue en 1224.8 minutos, es decir, en 2 días y medio aproximadamente.
- Para el trabajo 3 (25 y 26), se estima su entrega en 1630.4 minutos, es decir, en 3 días y medio aproximadamente.

Se puede verificar que el resto de trabajos pendientes no fueron afectados en su fecha de entrega por el ingreso de los nuevos trabajos. Estos tiempos máximos de entrega fueron definidos iguales a 1440 minutos para todos los trabajos pendientes, que representan no más de tres días para su finalización

Capítulo 5

Conclusiones y Recomendaciones

A continuación se presentan las conclusiones del estudio y recomendaciones con base en las complicaciones encontradas a lo largo del trabajo de investigación.

5.1. Conclusiones

La heurística base no resultó satisfactoria, pues en pocos casos se llegó al óptimo, aún en aquellos casos en los cuales se tenían pocos trabajos. Esto se debe a que no se planteó una inicialización del algoritmo con un ordenamiento adecuado. Esto resultó, sin embargo, importante al introducir un algoritmo de ordenamiento aleatorio en la segunda heurística (Heurística OA).

Con la heurística aleatoria implementada, se encontraron resultados altamente satisfactorios, en donde en un 90 % aproximadamente de los casos simulados que el modelo lineal encontraba el valor optimo, la heurística OA también los encontraba. En los casos en donde el modelo lineal no lograba encontrar el valor optimo en un tiempo prudente, la heurística OA encontraba un makespan menor en un tiempo también menor. Por lo que resultó adecuado implementar las heurísticas para la estimación de tiempos de entrega en una empresa manufacturera.

Cuando la instancia presenta una gran cantidad de trabajos y máquinas la heurística base puede ser sufuciente para obtener una mejor estimación de tiempos de entrega que el modelo lineal. En estos casos la heurística OA puede que encuentre una mejor solución, y en general lo encuentra, sin embargo el tiempo de ejecución es mayor.

En la heurística OA a mayor cantidad de iteraciones es más probable que encuentre una mejor solución, pero a coste de más tiempo de procesamiento.

Para la empresa Bórdalo se implementaron los tres algoritmos para simular una semana de trabajo, en donde se obtuvo que la huerística OA obtuvo el menor valor de makespan en un tiempo aceptable en comparación con el modelo lineal, por lo que se puede aplicar para realizar la estimación de los tiempos de entrega de los pedidos.

Aumentando restricciones como tiempo inicial por máquina y un tiempo máximo de entrega por trabajo se logra cubrir las necesidades de la empresa Bordalo para estimar nuevos tiempos de procesamiento de tal forma que, si ingresan nuevos trabajos en un tiempo determinado t, los trabajos anteriores no se atrasen y los nuevos trabajos estén listos lo antes posible. Por lo que la empresa puede darles tiempos de entrega a los clientes. De esta misma manera, al utilizar estas restricciones, se puede correr el algoritmo conociendo las fechas máximas que el cliente espera recibir el producto y obtener un horario de trabajo para las máquinas y productos.

Debido a que ciertos pedidos pueden pasar por solo un tipo de máquina, por ejemplo el cliente solo desea el trabajo de bordado de una prenda y no la elaboración del producto. Las heurísticas se pueden adaptar a este tipo de eventos al definir los tiempos de trabajo de ese pedido en los otros tipos de máquina iguales a cero.

Las heurísticas implementadas con las restricciones de tiempo inicial y tiempo de entrega permiten cubrir todas las necesidades que se pueden presentar dentro de una empresa manufacturera. Estos tiempos iniciales por máquina no solo pueden representar tiempos de trabajos previos por finalizar, sino también, tiempos de mantenimiento de las máquinas, entre otras cosas.

5.2. Recomendaciones

En la heurística aleatoria se sugiere implementar algoritmos de selección tales como los genéticos; ya que estos permitirían, mediante un criterio adecuado, ir eliminando las inicializaciones inadecuadas que generan un makespan alto en favor de aquellas que generen un makespan más cercano al optimo. De esta forma la inicialización de una nueva iteración no será aleatoria.

Las heurísticas pueden ser acomodadas de tal forma que, cuando existan trabajos que requieran ser procesados por solo un tipo de maquina, este no realize cálculos en los otros tipos de máquinas permitiendo así disminuir el tiempo de procesamiento

en el caso de que se presenten trabajos como los mencionados anteriormente.

Los algoritmos heurísticos implemetados estan programados de tal forma que solo realiza una tarea a la vez. Debido a la naturaleza iterativa del algoritmo, se pude mejorar paralelizando los cálculos y usando la capacidad multi-tarea de los procesadores y tarjetas gráficas modernas. De esta forma se reducirían los tiempos de cálculo significativamente.

En cuanto a la toma de datos, convendría implementar el modelo con volúmenes reales, así se podría comparar los resultados del modelo con la realidad. Debido a temas de confidencialidad la empresa prefirió no dar sus volúmenes de producción pues se podría obtener directamente sus niveles de ingresos reales.

Se considera importante poder establecer un sistema informático que facilite el uso del modelo para las empresas, así el modelo resultaría útil a nivel laboral.

Convendría ampliar la metodología a mayor número de procesos tales como los del caso de estudio, es decir, que un cierto trabajo tenga que ser procesado por más de dos tipos distintos de máquinas. Esta implementación necesariamente debe aplicarse con la paralelización de los cálculos, debido a que el tiempo de procesamiento aumentaría notablemente.

En pos de futuras y mejores aplicaciones del modelo, se plantea la sincronización del modelo con una base de tiempos de la propia empresa en continuo crecimiento y con la facilidad de incrementar el número de máquinas de acuerdo con el crecimiento propio de la empresa. De esta manera el modelo se puede ir reajustando de acuerdo con los nuevos tiempos ingresados y se tendría información actualizada para un control de producción interno.

Apéndice A

Apéndice

A.1. Tablas de tiempos

A continuación, las observaciones de los tiempos con los cuales se establecieron las distribuciones y se tomaron los promedios para realizar las simulaciones y los cálculos.

Cuadro A.1: Tiempos de los trabajos en gorras y bordados pequeños en minutos

	Bordado en gorras		Bordado pequeño	
	Menos de 3000 puntadas	Más de 3000 puntadas	Menos de 3000 puntadas	Entre 3000 y 6000 puntadas
1	4,79	7,22	3,31	5,02
2	4,77	8,04	2,92	4,89
3	4,76	7,91	3,33	4,92
4	4,76	9,18	3,57	4,83
5	4,80	9,42	3,09	4,69
6	4,71	9,75	3,40	5,23
7	4,79	7,51	3,14	4,59
8	4,85	7,96	3,17	4,98
9	4,87	6,82	3,18	4,64
10	4,86	9,98	3,13	4,92
11	4,81	7,19	2,95	4,85
12	4,77	7,76	3,35	5,01
13	4,79	8,73	3,16	4,89
14	4,78	8,59	2,85	4,85
15	4,83	7,95	3,23	5,02
16	4,78	6,37	3,13	5,02
17	4,79	6,82	2,75	4,83
18	4,77	7,01	3,18	5,09
19	4,83	9,04	2,89	4,93
20	4,78	10,58	2,95	5,09
21	4,75	8,72	3,01	5,10
22	4,86	11,07	3,23	5,28
23	4,76	8,27	3,37	5,01
24	4,92	8,33	3,26	4,88
25	4,82	8,37	3,10	4,67
26	4,73	8,20	3,27	4,76
27	4,84	8,40	3,23	5,01
28	4,82	7,92	3,03	4,77
29	4,88	10,30	2,92	5,18
30	4,74	8,41	3,16	4,99
31	4,86	7,31	2,69	5,12
32	4,87	8,83	3,38	5,03

Cuadro A.2: Tiempos de los trabajos en bordados medianos y grandes en minutos

	Bordado mediano		Bordado grande	
	Entre 6000 y 9000 puntadas	Entre 9000 y 12000 puntadas	Entre 12000 y 15000 puntadas	Más de 15000 puntadas
1	6,75	10,25	15,13	19,06
2	6,79	10,05	15,27	19,49
3	7,00	10,06	15,27	18,93
4	7,40	10,07	15,23	18,97
5	7,35	9,95	15,23	18,99
6	7,32	9,92	15,20	19,57
7	6,65	10,06	15,40	19,41
8	6,83	10,13	15,13	19,16
9	6,43	9,97	15,27	19,13
10	6,97	10,00	15,26	19,22
11	7,65	10,08	15,33	18,84
12	7,18	10,12	15,32	19,43
13	6,82	9,90	15,30	19,45
14	7,24	9,94	15,26	18,89
15	7,20	10,16	15,15	19,35
16	7,43	9,98	15,37	19,17
17	7,00	10,06	15,13	19,17
18	6,74	10,04	15,36	19,21
19	7,05	10,12	15,21	19,01
20	6,90	10,11	15,18	18,96
21	7,09	10,02	15,22	19,03
22	6,66	10,15	15,14	18,93
23	6,86	10,10	15,19	19,29
24	6,74	10,11	15,28	18,94
25	6,89	10,15	15,25	19,34
26	7,02	9,95	15,32	19,38
27	7,51	9,99	15,23	19,00
28	7,94	10,20	15,08	19,15
29	6,95	10,00	15,26	19,18
30	6,66	10,12	15,25	19,31
31	7,16	9,93	15,27	19,26
32	6,83	10,07	15,23	19,02

A.2. Códigos

A continuación, se presentan los códigos implementados en *python*.

A.2.1. Modelo Lineal

```
from gurobipy import *
import numpy as np
import pandas as pd
import random
try:
m = Model("Modelo1")
bordado  = ["A","B","C"]
estampado = ["M","N","O"]
maquinas = bordado + estampado
trabajos  = list(range(1,41))
traux     = list(range(0,41))
resultados_m1 = []

np.random.seed(200)
tiempo_trabajos = np.random.uniform(low=2, high=10, size=(len(maquinas),
len(trabajos)))
tiempo_trabajos = tiempo_trabajos.round(1)

tiempos = pd.DataFrame(tiempo_trabajos, index = bordado + estampado,
columns = trabajos)
tiempo = {}
for i in bordado + estampado:
for j in trabajos:
tiempo[(i,j)] = tiempos.loc[i, j]

# Crear variables para el modelo
x = {}
for i in maquinas:
for j in traux:
for k in trabajos:
```

```
if j != k :
x[i,j,k] = m.addVar(vtype=GRB.BINARY,
name="x("+i+","+str(j)+","+str(k)+")")
c = {}
for i in maquinas :
for k in traux :
c[i,k] = m.addVar(vtype=GRB.CONTINUOUS,lb = 0, ub = GRB.INFINITY,
name="c("+i+",
"+str(k)+")")

cmax = m.addVar(vtype=GRB.CONTINUOUS,lb = 0, ub = GRB.INFINITY,
name="c")

# Funcion objetivo
m.setObjective(cmax, GRB.MINIMIZE)

# Restricciones
m.addConstrs(
(quicksum(x[i,j,k] for i in maquinas for j in traux if j!=k ) == 1
for k in trabajos),"c1")

m.addConstrs(
(quicksum(x[i,j,k] for i in bordado for j in traux if j!=k ) == 0
for k in trabajos if k % 2 == 0),"c1.1")

m.addConstrs(
(quicksum(x[i,j,k] for i in estampado for j in traux if j!=k ) == 0
for k in trabajos if k % 2 != 0),"c1.2")

m.addConstrs(
(quicksum(c[i,j] for i in maquinas) +
quicksum(x[k,l,j+1]*tiempo[k,j+1]
for l in traux if l != (j+1)) <= c[k,j+1] for k in maquinas for j
in trabajos if j % 2 != 0),"c1.3")
```

```
m.addConstrs(
(quicksum(x[i,j,k] for i in maquinas for k in trabajos if j!=k )
<= 1 for j in trabajos),"c2")

m.addConstrs((quicksum(x[i,0,k] for k in trabajos ) <= 1
for i in maquinas),"c3")

m.addConstrs(
(quicksum(x[i,h,j] for h in traux if (j!=h and k!=h) ) >= x[i,j,k]
for j in trabajos for k in trabajos for i in maquinas
if j!=k),"c4")

m.addConstrs(
(c[i,k] + 100000*(1-x[i,j,k]) >= c[i,j] + tiempo[i,k]
for i in maquinas for j in traux for k in trabajos if j!=k),"c5")

m.addConstrs(
(c[i,0] == 0  for i in maquinas),"c6")

m.addConstrs(
(c[i,j] >= 0  for j in trabajos for i in maquinas),"c7")

m.addConstrs(
(c[i,j] <= cmax  for j in trabajos for i in maquinas),"c8")

m.optimize()

for v in m.getVars():
if v.x > 0:
print('%s %g' % (v.varName, v.x))
resultados_m1.append([v.varName,v.x])
print('Obj: %g' % m.objVal)

except GurobiError as e:
print('Error code ' + str(e.errno) + ": " + str(e))
```

```
except AttributeError:
print('Encountered an attribute error')
```

A.2.2. Heurística Base

```
import numpy as np
import pandas as pd
import operator
from copy import deepcopy
import random
import timeit

bordado  = ["A","B","C"]
estampado = ["M","N","O"]
maquinas = bordado + estampado
trabajos  = list(range(1,41))

np.random.seed(200)
tiempo_trabajos = np.random.uniform(low=2, high=10,
size=(len(maquinas),len(trabajos)))
tiempo_trabajos = tiempo_trabajos.round(1)
tiempos = pd.DataFrame(tiempo_trabajos, index = bordado +
estampado, columns = trabajos)

tiempo = {}
for i in bordado + estampado:
for j in trabajos:
tiempo[(i,j)] = tiempos.loc[i, j]

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
and i[1] % 2 != 0):
tiempo[i] = 0

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
```

```
and i[1] % 2 != 0):
tiempo[i] = 0

start = timeit.default_timer()
maximos = [max([tiempo.get(key) for key in [t for t in tiempo
if t[1] == m]]) for m in range(1,len(trabajos)+1)]
l = [round(maximos[i] + maximos[i+1],2)
for i in np.arange(len(trabajos)) if i % 2 == 0 ]
tiempos_totales = dict(zip([i for i in range(1,len(trabajos)+1)
if i % 2 != 0 ], l))
tiempos_ordenados = sorted(tiempos_totales.items(),
key=operator.itemgetter(1), reverse = True)

#crear objetos que almacenaran toda la información
horario = [ [] for i in maquinas ]
#genero la lista de listas para la heurística
trabajos_tiempof = dict()

#se pone el primer trabajo en las posiciones tal que genere
#el menor tiempo de makespan

trab = tiempos_ordenados[0][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
indice1 = aux1.index(np.nanmin(aux1))
indice2 = aux2.index(np.nanmin(aux2))
valor1 = np.nanmin(aux1)
valor2 = np.nanmin(aux2)

horario[indice1].insert(0,[0,trab,0,valor1])
horario[indice2].insert(0,[0,trab+1,valor1,valor1 + valor2])

horario_final =deepcopy(horario)
trabajos_tiempof[trab] = valor1
```

```
trabajos_tiempof[trab+1] = valor1 + valor2
tiempo_final = trabajos_tiempof.copy()

#proceso completo de iteraciones
for p in np.arange(1,len(tiempos_ordenados)) :
trab = tiempos_ordenados[p][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
maxi = 1000000
horario = deepcopy(horario_final)
trabajos_tiempof = tiempo_final.copy()
for i in np.arange(0,len(bordado)) :
iteracion = len(horario[i])
for k in np.arange(0,iteracion+1) :
horario_aux = deepcopy(horario)
tiempo_trabajo = trabajos_tiempof.copy()
for v in np.arange(k,iteracion+1) :
if v != k :
tiempo_aux = (horario_aux[i][v][3] - horario_aux[i][v][2]).copy()
horario_aux[i][v][2] = horario_aux[i][v-1][3].copy()
horario_aux[i][v][3] = horario_aux[i][v][2].copy() + tiempo_aux
horario_aux[i][v][0] = horario_aux[i][v-1][1]
tiempo_trabajo[horario_aux[i][v][1]] = horario_aux[i][v][3].copy()
elif v == 0 :
horario_aux[i].insert(0,[0,trab,0,aux1[i]])
temp = aux1[i].copy()
tiempo_trabajo[trab] = temp.copy()
elif v != 0 :
horario_aux[i].insert(v,[horario_aux[i][v-1][1],trab,
horario_aux[i][v-1][3],horario_aux[i][v-1][3] + aux1[i]])
temp = horario_aux[i][v][3].copy()
tiempo_trabajo[trab] = temp.copy()
horario_aux2 = deepcopy(horario_aux)
tiempo_trabajo2 = tiempo_trabajo.copy()
```

```
for j in np.arange(len(bordado),len(bordado)+len(estampado)):
it = len(horario[j])
for m in np.arange(0,it+1) :
horario_aux = deepcopy(horario_aux2)
tiempo_trabajo = tiempo_trabajo2.copy()
if m == 0 :
horario_aux[j].insert(0,[0,trab+1,tiempo_trabajo[trab],
tiempo_trabajo[trab]+aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
elif m != 0 :
if horario_aux[j][m-1][3] <= tiempo_trabajo[trab] :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
tiempo_trabajo[trab],tiempo_trabajo[trab] + aux2[j]])
else :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
horario_aux[j][m-1][3],horario_aux[j][m-1][3] + aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
for l in [w for w in np.arange(len(bordado),len(bordado)+
len(estampado)) if len(horario_aux[w]) > 0 ]:
for n in np.arange(0,len(horario_aux[l])) :
tiempo_aux = horario_aux[l][n][3] - horario_aux[l][n][2].copy()
if n != 0 :
if horario_aux[l][n-1][3] <= tiempo_trabajo[
horario_aux[l][n][1]-1] :
horario_aux[l][n][2] =
tiempo_trabajo[horario_aux[l][n][1]-1].copy()
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]
else :
horario_aux[l][n][2] = horario_aux[l][n-1][3].copy()
horario_aux[l][n][3] = horario_aux[l][n-1][3] + tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]
tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
```

```
elif n==0:
horario_aux[l][n][2] = tiempo_trabajo[horario_aux[l][n][1]-1]
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux
tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
if maxi > max(tiempo_trabajo.values()) :
maxi = max(tiempo_trabajo.values()).copy()
horario_final = deepcopy(horario_aux)
tiempo_final = tiempo_trabajo.copy()

stop = timeit.default_timer()
print('Time: ', stop - start)
print('cmax: ', maxi)
```

A.2.3. Heurística OA

```
import numpy as np
import pandas as pd
import operator
from copy import deepcopy
import timeit
bordado  = ["A","B","C"]
estampado = ["M","N","O"]
maquinas = bordado + estampado
trabajos  = list(range(1,41))

np.random.seed(200)
tiempo_trabajos = np.random.uniform(low=2, high=10,
size=(len(maquinas),len(trabajos)))
tiempo_trabajos = tiempo_trabajos.round(1)

tiempos = pd.DataFrame(tiempo_trabajos, index = bordado +
estampado, columns = trabajos)

tiempo = {}
for i in bordado + estampado:
```

```
for j in trabajos:
tiempo[(i,j)] = tiempos.loc[i, j]

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
and i[1] % 2 != 0):
tiempo[i] = 0

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
and i[1] % 2 != 0):
tiempo[i] = 0

start = timeit.default_timer()
maximos = [max([tiempo.get(key) for key in [t for t in tiempo
if t[1] == m]]) for m in range(1,len(trabajos)+1)]
l = [round(maximos[i] + maximos[i+1],2) for i in
np.arange(len(trabajos)) if i % 2 == 0 ]
tiempos_totales = dict(zip([i for i in range(1,len(trabajos)+1)
if i % 2 != 0 ], l))
tiempos_ordenados = sorted(tiempos_totales.items(),
key=operator.itemgetter(1), reverse = True)

#crear objetos que almacenaran toda la información

maxi_intentos = 1000000
for intentos in np.arange(30):
np.random.seed(None)
np.random.shuffle(tiempos_ordenados)
horario = [ [] for i in maquinas ]
trabajos_tiempof = dict()
trab = tiempos_ordenados[0][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
```

```
indice1 = aux1.index(np.nanmin(aux1))
indice2 = aux2.index(np.nanmin(aux2))
valor1 = np.nanmin(aux1)
valor2 = np.nanmin(aux2)

horario[indice1].insert(0,[0,trab,0,valor1])
horario[indice2].insert(0,[0,trab+1,valor1,valor1 + valor2])

horario_final =deepcopy(horario)
trabajos_tiempof[trab] = valor1
trabajos_tiempof[trab+1] = valor1+valor2
tiempo_final = trabajos_tiempof.copy()
#proceso completo de iteraciones
for p in np.arange(1,len(tiempos_ordenados)) :
trab = tiempos_ordenados[p][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
maxi = 1000000
horario = deepcopy(horario_final)
trabajos_tiempof = tiempo_final.copy()
for i in np.arange(0,len(bordado)) :
iteracion = len(horario[i])
for k in np.arange(0,iteracion+1) :
horario_aux = deepcopy(horario)
tiempo_trabajo = trabajos_tiempof.copy()
for v in np.arange(k,iteracion+1) :
if v != k :
tiempo_aux = (horario_aux[i][v][3] - horario_aux[i][v][2]).copy()
horario_aux[i][v][2] = horario_aux[i][v-1][3].copy()
horario_aux[i][v][3] = horario_aux[i][v][2].copy() + tiempo_aux
horario_aux[i][v][0] = horario_aux[i][v-1][1]
tiempo_trabajo[horario_aux[i][v][1]] = horario_aux[i][v][3].copy()
elif v == 0 :
horario_aux[i].insert(0,[0,trab,0,aux1[i]])
```

```
temp = aux1[i].copy()
tiempo_trabajo[trab] = temp.copy()
elif v != 0 :
horario_aux[i].insert(v,[horario_aux[i][v-1][1],trab,
horario_aux[i][v-1][3],horario_aux[i][v-1][3] + aux1[i]])
temp = horario_aux[i][v][3].copy()
tiempo_trabajo[trab] = temp.copy()
horario_aux2 = deepcopy(horario_aux)
tiempo_trabajo2 = tiempo_trabajo.copy()
for j in np.arange(len(bordado),len(bordado)+len(estampado)):
it = len(horario[j])
for m in np.arange(0,it+1) :
horario_aux = deepcopy(horario_aux2)
tiempo_trabajo = tiempo_trabajo2.copy()
if m == 0 :
horario_aux[j].insert(0,[0,trab+1,tiempo_trabajo[trab],
tiempo_trabajo[trab]+aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
elif m != 0 :
if horario_aux[j][m-1][3] <= tiempo_trabajo[trab] :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
tiempo_trabajo[trab],tiempo_trabajo[trab] + aux2[j]])
else :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
horario_aux[j][m-1][3],horario_aux[j][m-1][3] + aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
for l in [w for w in np.arange(len(bordado),
len(bordado)+len(estampado)) if
len(horario_aux[w]) > 0 ]:
for n in np.arange(0,len(horario_aux[l])) :
tiempo_aux = horario_aux[l][n][3].copy() -
horario_aux[l][n][2].copy()
if n != 0 :
if horario_aux[l][n-1][3] <= tiempo_trabajo[
```

```
horario_aux[l][n][1]-1] :
horario_aux[l][n][2] =
tiempo_trabajo[horario_aux[l][n][1]-1].copy()
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]
else :
horario_aux[l][n][2] = horario_aux[l][n-1][3].copy()
horario_aux[l][n][3] = horario_aux[l][n-1][3] + tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]
tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
elif n==0:
horario_aux[l][n][2] = tiempo_trabajo[horario_aux[l][n][1]-1]
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux
tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
if maxi > max(tiempo_trabajo.values()) :
maxi = max(tiempo_trabajo.values()).copy()
horario_final = deepcopy(horario_aux)
tiempo_final = tiempo_trabajo.copy()

if maxi_intentos>maxi:
horario_intentos = deepcopy(horario_final)
tiempo_intentos = tiempo_final.copy()
maxi_intentos = maxi.copy()
stop = timeit.default_timer()
print('Time: ', stop - start)
print('cmax: ', maxi_intentos)
```

A.2.4. Modificación de Heurística de Ordenamiento Aleatorio para tiempos iniciales

```
import numpy as np
import pandas as pd
import operator
from copy import deepcopy
```

```
import timeit

bordado  = ["A","B","C"]
estampado = ["D","E","F"]
maquinas = bordado + estampado
trabajos  = list(range(1,21))

np.random.seed(200)
tiempo_trabajos = np.random.uniform(low=2, high=10,
size=(len(maquinas),len(trabajos)))
tiempo_trabajos = tiempo_trabajos.round(1)

tiempo = {}
for i in bordado + estampado:
for j in trabajos:
tiempo[(i,j)] = tiempos.loc[i, j]

tiempo_inicial = [0,0,0,0,0,0]

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
and i[1] % 2 != 0):
tiempo[i] = 0

for i in tiempo.keys():
if (i[0] in bordado and i[1] % 2 == 0) or (i[0] in estampado
and i[1] % 2 != 0):
tiempo[i] = 0

start = timeit.default_timer()
maximos = [max([tiempo.get(key) for key in [t for t in tiempo
if t[1] == m]]) for m in range(1,len(trabajos)+1)]
l = [round(maximos[i] ,2) for i in np.arange(len(trabajos))
if i % 2 == 0 ]
tiempos_totales = dict(zip([i for i in range(1,len(trabajos)+1)
if i % 2 != 0 ], l))
```

```
tiempos_ordenados = sorted(tiempos_totales.items(),
key=operator.itemgetter(1), reverse = True)

#crear objetos que almacenaran toda la información
horario = [ [] for i in maquinas ]
#genero la lista de listas para la heurística
trabajos_tiempof = dict()
#se pone el primer trabajo en las posiciones tal que genere
#el menor tiempo de makespan

np.random.seed(None)
np.random.shuffle(tiempos_ordenados)

trab = tiempos_ordenados[0][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
indice1 = aux1.index(np.nanmin(aux1))
indice2 = aux2.index(np.nanmin(aux2))
valor1 = np.nanmin(aux1)
valor2 = np.nanmin(aux2)



horario[indice1].insert(0,[0,trab,tiempo_inicial[indice1],
valor1+tiempo_inicial[indice1]])

if horario[indice1][0][3]  < tiempo_inicial[indice2] :
horario[indice2].insert(0,[0,trab+1,
tiempo_inicial[indice2],tiempo_inicial[indice2] + valor2])
else :
horario[indice2].insert(0,[0,trab+1,
horario[indice1][0][3],horario[indice1][0][3] + valor2])

horario_final =deepcopy(horario)
trabajos_tiempof[trab] = valor1+tiempo_inicial[indice1]
```

```
trabajos_tiempof[trab+1] = horario[indice2][0][3]
tiempo_final = trabajos_tiempof.copy()

#proceso completo de iteraciones
for p in np.arange(1,len(tiempos_ordenados)) :
trab = tiempos_ordenados[p][0]
aux1 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab]]
aux2 = [tiempo.get(key) for key in [t for t in tiempo
if t[1] == trab+1]]
maxi = 1000000
horario = deepcopy(horario_final)
trabajos_tiempof = tiempo_final.copy()
for i in np.arange(0,len(bordado)) :
iteracion = len(horario[i])
for k in np.arange(0,iteracion+1) :
horario_aux = deepcopy(horario)
tiempo_trabajo = trabajos_tiempof.copy()
for v in np.arange(k,iteracion+1) :
if v != k :
tiempo_aux = (horario_aux[i][v][3] - horario_aux[i][v][2]).copy()
horario_aux[i][v][2] = horario_aux[i][v-1][3].copy()
horario_aux[i][v][3] = horario_aux[i][v][2].copy() + tiempo_aux
horario_aux[i][v][0] = horario_aux[i][v-1][1]
tiempo_trabajo[horario_aux[i][v][1]] = horario_aux[i][v][3].copy()
elif v == 0 :
horario_aux[i].insert(0,[0,trab,tiempo_inicial[i],aux1[i]+
tiempo_inicial[i]])
temp = (aux1[i]+tiempo_inicial[i]).copy()
tiempo_trabajo[trab] = temp.copy()
elif v != 0 :
horario_aux[i].insert(v,[horario_aux[i][v-1][1],trab,
horario_aux[i][v-1][3],horario_aux[i][v-1][3] + aux1[i]])
temp = horario_aux[i][v][3].copy()
tiempo_trabajo[trab] = temp.copy()
horario_aux2 = deepcopy(horario_aux)
```

```
tiempo_trabajo2 = tiempo_trabajo.copy()
for j in np.arange(len(bordado),len(bordado)+len(estampado)):
it = len(horario[j])
for m in np.arange(0,it+1) :
horario_aux = deepcopy(horario_aux2)
tiempo_trabajo = tiempo_trabajo2.copy()
if m == 0 :
if tiempo_trabajo[trab] < tiempo_inicial[j] :
horario_aux[j].insert(0,[0,trab+1,tiempo_inicial[j],
tiempo_inicial[j]+aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
else :
horario_aux[j].insert(0,[0,trab+1,tiempo_trabajo[trab],
tiempo_trabajo[trab]+aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
elif m != 0 :
if horario_aux[j][m-1][3] <= tiempo_trabajo[trab] :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
tiempo_trabajo[trab],tiempo_trabajo[trab] + aux2[j]])
else :
horario_aux[j].insert(m,[horario_aux[j][m-1][1],trab+1,
horario_aux[j][m-1][3],horario_aux[j][m-1][3] + aux2[j]])
temp = horario_aux[j][m][3].copy()
tiempo_trabajo[trab+1] = temp.copy()
for l in [w for w in
np.arange(len(bordado),len(bordado)+len(estampado)) if
len(horario_aux[w]) > 0 ]:
for n in np.arange(0,len(horario_aux[l])) :
tiempo_aux = horario_aux[l][n][3] - horario_aux[l][n][2]
if n != 0 :
if horario_aux[l][n-1][3] <=
tiempo_trabajo[horario_aux[l][n][1]-1] :
horario_aux[l][n][2] =
tiempo_trabajo[horario_aux[l][n][1]-1].copy()
```

```
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]
else :
horario_aux[l][n][2] = horario_aux[l][n-1][3].copy()
horario_aux[l][n][3] = horario_aux[l][n-1][3] + tiempo_aux
horario_aux[l][n][0] = horario_aux[l][n-1][1]

tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
elif n==0:
if tiempo_trabajo[horario_aux[l][n][1]-1] < tiempo_inicial[l] :
horario_aux[l][n][2] = tiempo_inicial[l]
horario_aux[l][n][3] = tiempo_inicial[l] + tiempo_aux

tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
else :
horario_aux[l][n][2] = tiempo_trabajo[horario_aux[l][n][1]-1]
horario_aux[l][n][3] = tiempo_trabajo[horario_aux[l][n][1]-1] +
tiempo_aux

tiempo_trabajo[horario_aux[l][n][1]] = horario_aux[l][n][3].copy()
if maxi > max(tiempo_trabajo.values()) :
maxi = max(tiempo_trabajo.values()).copy()
horario_final = deepcopy(horario_aux)
tiempo_final = tiempo_trabajo.copy()

stop = timeit.default_timer()
print('Time: ', stop - start)
print('cmax: ', maxi)
```

Bibliografía

[1] L. FANJUL-PEYRÓ AND R. RUIZ, *Iterated greedy local search methods for unrelated parallel machine scheduling*, European Journal of Operational Research, 207 (2010), p. 55–69.

[2] G. R. FRAMINAN J, LEISTEN R, *Manufacturing scheduling systems: An integrated view on models, methods and tools*, Springer, London, 2014.

[3] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York, 1979.

[4] U. R. S. S. Y. K. K. GAREY, K., *Measuring the quality of manufacturing schedules*, Intelligent Scheduling Systems, 4 (1995), p. 129–254.

[5] J. N. D. GUPTA AND R. A. DUDEK, *Optimality criteria for flowshop schedules*, IIE Transactions, 3 (1971), p. 199–205.

[6] R. K. A. H. G. LENSTRA, J. K. AND P. BRUCKER, *Complexity of machine scheduling problems*, Annals of Discrete Mathematics, 1 (1977), p. 343–362.

[7] S. D. B. LENSTRA, J. K. AND E. TARDOS, *Approximation algorithms for scheduling unrelated parallel machines*, Mathematical Programming, 46 (1990), p. 259–271.

[8] E. J. Y. H. I. NAWAZ, M., *A heuristic algorithm for the n-job flow-shop sequencing problem*, The International Journal of Management Science, 11 (1983), p. 91–95.

[9] M. L. PINEDO, *Scheduling: Theory, Algorithms, and Systems*, Springer, New York, 2012.

[10] S. J. E. F. TSENG, F. T. AND J. N. D. GUPTA, *An empirical analysis of integer programming formulations for the permutation flowshop*, OMEGA, The International Journal of Management Science, 32 (2004), p. 285–293.

[11] E. VALLADA AND R. RUIZ, *Genetic algorithms for the unrelated parallel machine scheduling problem with sequence dependent setup times*, European Journal of Operational Research, 211 (2011), p. 612–622.

ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

CARRERA DE INGENIERÍA MATEMÁTICA

ORDEN DE EMPASTADO

De acuerdo a lo estipulado en el artículo 27 del Instructivo para la Implementación de la Unidad de Titulación en las Carreras y Programas vigentes de la Escuela Politécnica Nacional, aprobado por Consejo de Docencia en sesión extraordinaria del 29 de abril de 2015 y una vez verificado el cumplimiento del formato de presentación establecido, se autoriza la impresión y encuadernación final del Trabajo de Titulación presentado por los señores **LUIS SEBASTIÁN ALCÍVAR RODRÍGUEZ** y **RAFAEL ALEJANDRO CHÁVEZ RIERA**.

Fecha de autorización: Quito, D.M., 31 de julio de 2019.

Dr. Polo Vaca
DECANO
FACULTAD DE CIENCIAS