Modelo para la Programación de la Producción, por medio de Algoritmo Genético, Métodos Heurísticos y Reglas de Despacho en Concreaceros SAS.

Juan Manuel Bitar Polo

.

Corporación Universitaria Del Caribe - CECAR

Facultad De Ciencias Básicas, Ingenierías Y Arquitectura

Programa De Ingeniería Industrial

Sincelejo.

2016

Modelo para la Programación de la Producción, por medio de Algoritmo Genético, Métodos Heurísticos y Reglas de Despacho en Concreaceros SAS.

Juan Manuel Bitar Polo

Trabajo de Grado para Optar al Título de Ingeniero Industrial

Director

Ing. Rafael Merlano Porto

Magister en Ingeniería Industrial

Corporación Universitaria Del Caribe - CECAR

Facultad De Ciencias Básicas, Ingenierías Y Arquitectura

Programa De Ingeniería Industrial

Sincelejo.

2016

Nota de Aceptación

4.7

Firma presidente del jurado

Firma del jurado

Firma del jurado

Sincelejo, Sucre, 20 de Octubre de 2016

Tabla de Contenido

Listado de tablas

Listado de figuras

Resumen

La presente investigación se fundamenta principalmente en un modelo para programar la producción de una manera eficiente, con respecto a los tiempos de terminación de los lotes a fabricar en Concreaceros S.A.S. ya que se presenta de una manera Sistemática el incumplimiento en los tiempos de entrega, resultando con ello la insatisfacción de los clientes. Por tal motivo, dicho modelo se ajusta a cada célula de trabajo contribuyendo al mejoramiento de su rendimiento, con base en lo anterior se presenta como tema de investigación: la programación de la producción en dicha empresa para así comprobar la aceptabilidad de aplicar el modelo propuesto, teniendo como función objetivo el minimizar el tiempo de terminación máximo (*Makespan*). Para solucionar este problema se ha diseñado un modelo basado en un algoritmo genético modificados bajo un ambiente de máquinas en paralelo idénticas y la heurística de Johnson. Además, se precisa la adopción de reglas de despacho para poder tener en cuenta criterios evaluativos para considerar y escoger la secuencia acertada para ir en pro a la satisfacción de los clientes, y en la toma de decisiones basada en datos concretos.

***Palabras clave*:** Programación de la producción (scheduling), single machine, algoritmo genético, heurística de Johnson, reglas de despacho y Makespan.

Abstract

This research is primarily based on a model to schedule production efficiently, regarding completion times for batch manufacturing in Concreaceros S.A.S. as it is presented in a systematic failure in delivery times, thereby resulting in customer dissatisfaction. Therefore, this model fits each work cell helping to improve performance, based on the foregoing it is presented as a topic of research: production scheduling in the company in order to verify the acceptability of applying the model proposed, aiming at minimizing the role the maximum completion time (*Makespan*). To solve this problem based on a genetic algorithm modified under an atmosphere of identical machines in parallel and Johnson heuristic model is designed. Moreover, the adoption of rules dispatch is required to take into account evaluation criteria to consider and choose the correct sequence to go pro to customer satisfaction, and decision-making based on facts.

Keywords: Production scheduling, single machine, genetic algorithm, heuristic Johnson, decision rules and Makespan.

Introducción

En el presente proyecto de investigación, como su nombre lo refiere, el diseño de un modelo ajustado a las necesidades de la empresa tomada para la realización del presente estudio. Concreaceros S.A.S, presenta actualmente el parcialmente cumplimiento de las ordenes emitidas por los clientes, debido a la errónea asignación de tareas en los respectivos puesto de trabajos, a esto se le suman la ineficiencia en la planificación de los procesos y bajos controles de producción, aumentando con ello el bajo cumplimento de las fechas de entrega de los lotes a fabricar.

Por tal motivo, este modelo cuenta con el diseño que permita la programación de la producción, es decir; la asignación de tareas en las respetivas células de trabajo, teniendo como finalidad disminuir el tiempo de terminación máximo de la remisión (*Makespan*), trayendo como resultado la entrega oportuna de las órdenes fabricadas. Definiéndose con ello las variables y parámetros pertinentes para el diseño: del algoritmo genético ajustado a la asignación de tareas bajo un ambiente de máquinas en paralelo idénticas, y la heurística de Johnson para la programación de tareas en el área de figuración estándar: logrando con ello procesos eficientes de programación; para la disminución de los tiempos de producción para el primer mencionado y de los tiempos muertos o improductivos para el segundo mencionado. Además, se diseñó para el área de figuración estándar dos heurísticas que permita la obtención de pasos confiables y certeros para la elaboración de tareas en esta célula de trabajo, debido a la falta de procesos no automatizados y la alta intervención del operario.

Y en último lugar, el diseño de algunas reglas de despachos para la organización y planificación de la mano de obra y tiempo de procesamiento estimado: para las remisiones en cola, todo esto contrastado con las fechas de entregas respectivas y con la ayuda de criterios evaluativos, que permita valorar de manera predeterminada: el orden con que mejor se ajuste a los requerimientos actuales de la empresa. Todo esto se elaboró y diseño con la ayuda de Matlab®, validado con el software de programación: Lenkin®. permitiendo con todo esto medir su funcionalidad e idoneidad para el sistema de producción en Concreaceros S.A.S.

Planteamientos de la problemática

Es de suma importancia resaltar, que la relación o canal de comercialización de Concreaceros S.A.S. y sus clientes, es directa, es decir: no hay intermediarios en la recepción de las necesidades de los clientes y la respuesta de la empresa; por lo tanto, deja a la empresa en un nivel alto de compromiso y responsabilidad hacia sus clientes. Desafortunadamente los clientes en su totalidad no quedan satisfechos con respecto al tiempo de entrega de los productos, de manera sistemática se han evidenciado problemas relacionado con el incumplimiento de las ordenes de los clientes debido al atraso que se presentan en los tiempos de entrega, y la poca coordinación en los procesos de recepción, secuencia, elaboración y despacho de las órdenes de producción o de pedidos.

Partiendo de allí, se pudo observar que la asignación de las órdenes en los puestos de trabajos se hace de manera convencional o empírica, si un orden estructura o planificado de acuerdo a una regla de programación que optimice los tiempos de procesamiento y mejore el servicio al cliente, además la secuencia estimada de las respectivas órdenes a procesar en la planta deja un alto grado de incertidumbre, ya que carece de un proceso planificado de los procesos y programas de producción. En ese sentido no se reconoce la secuencia y orden de los trabajos a ejecutar, o de planes eficientes que den cuenta del comportamiento de variables y parámetros tales como: *Makespan* (tiempos de terminación máximo), números de trabajos adelantados, adelanto máximo, número de trabajos tardíos y tardanza máxima.

Otra de las falencias detectadas se encuentra en el área de figuración, ya que la demanda del mercado de acuerdo a las solicitudes y datos históricos es alta, es decir; los clientes piden grandes cantidades de acero estándar figurado, pero se suscita un problema en el ritmo de trabajo y realización del mismo, ya que estos son establecidos por el operario encargado, debido a que el proceso no está automatizado; por lo tanto, en los procesos es requerido un alto grado del trabajo del operario, debido a la falta de procesos automatizados o al bajo desarrollo de los procesos de transformación y sus métodos empíricos no son los mejores. Además, la no identificación del cuello de botella en dicha línea de producción hace que aumente su tiempo ocio y a su vez se

encuentren algunas estaciones con sobrecarga laboral, y por ende su proceso es deficiente con respecto al tiempo de ejecución, insumos utilizados, sobrantes y número de trabajos realizados en el tiempo estimado.

Los pocos controles en el área de producción pueden influenciar de manera negativa el cumplimiento de las ordenes asignadas en los puestos de trabajo, ya que el orden establecido juega un papel importante, y al ser alterado por los operarios genera un descontrol, que aumenta el índice de improductividad, sobre la cantidad total de trabajos requeridos en el tiempo de producción estimado previamente.

Con base en lo anterior, y a las teorías de programación es posible mejorar y efectuar una eficiente asignación de tareas en las diferentes maquinas o procesos y también a una correcta sincronización.

Justificación de la problemática

Unos de los aspectos a considerar en la gestión de la producción, es el término programación (*scheduling*), visto desde un punto de vista macro, consta de dos objetivos: el primero es poder dar respuesta a las solicitudes de los clientes, es decir procesar esa información de entrada y convertirla en productos, sea tangible o intangible, y como segundo objetivo es poder procesarlas en el menor tiempo posible (Almeida Rodríguez & Melián Batista, 2007). Con base en lo anterior, y a las teorías de programación es posible mejorar y efectuar una eficiente asignación de tareas en las diferentes maquinas o procesos y también a una correcta sincronización. Partiendo de allí se puede obviar que con una adecuada asignación de tareas a los respectivos puestos de trabajo; conlleva a una sincronización en los procesos, control de la producción y en las existencias de partes terminadas, y aun llegar a disminuir los atrasos y adelantos excesivos en las ordenes de los clientes, consiguiendo con esto un nivel adecuado de respuesta ante las fecha de entrega y un aumento en la eficiencia en lo que concierne a la utilización de la mano de obra y maquinaria.

En Concreaceros S.A.S, empresa tomada para el estudio en esta rama de la ingeniería, se ha notado el parcial cumplimiento en los tiempos de entregas, debido a que la forma de planificar, asignar y controlar las tareas en los respectivos puestos de trabajos no son los adecuados, puesto que asignan de una manera Empírica, sin modelo que planifique u optimice los trabajos a procesar sin su respectiva evaluación, dejando a un lado criterios como el tiempo de terminación máximo, número de trabajos tardíos, tardanza máxima, número de trabajos adelantados y adelanto máximo, factores que pueden servir de ayuda, ya que dicha permutación al ser evaluada según los criterios mencionados anteriormente. Los resultados que arroje el estudio podrán ser medidos y a su vez alterando el orden cambiará también en algunas condiciones los valores de los criterios, dejando claro que el cálculo predeterminados de ellos puede ayudar en la toma decisiones.

Con base en lo anterior se puede medir la complejidad del problema, ya que al aumentar el número de trabajos, se hace más complejos evaluar el número total de posibles combinaciones,

ya que al priorizar el orden, se hace indispensable la utilización de algoritmos, aunque no se llegue en algunos casos a recorrer todo el espacio muestral, si se podrá explorar un espacio considerable de búsqueda en el caso de los algoritmos genéticos y así poder reducir el nivel de incertidumbre, es decir; que ya se contará con diferentes combinaciones y se tendrá la libertad de escoger un óptimo local según la función objetivo definida, la mejor de las soluciones comparadas. Para aquellos proceso menos complejos se tiene a la mano heurísticas y método exactos que pueden servir de guía y notable ayuda en la evaluación de dichos criterios, por lo tanto se hace indispensable la definición del ambiente del trabajo junto con sus respectivas restricciones, y la manera de como divulgar la información en la planta, es decir; Promover por el cumplimiento y control de la puesta en marcha de lo programado, para así poder brindar soluciones adecuadas, que mitigan los desajustes en los tiempos de entregas.

Es preciso resaltar, que en aquellos puestos de trabajos donde el ritmo es impuesto por el operario, sea reevaluado, haciendo un estricto control y vigilando su proceso; y si es posible verificar si su metodología es la adecuada, para su postrera reestructuración, evitando que pueda incentivarse el descontrol en la producción y contribuir en cierto modo en los atrasos. Por lo tanto, se hace viable e indispensable el estudio investigativo para el diseño de un sistema que permita planificar, programar y controlar la producción en Concreaceros S.A.S.

Objetivos

Objetivo general

- Diseñar un programa de producción, mediante un algoritmo genético modificado y heurísticas, con la finalidad de disminuir los tiempos de procesamiento y de entrega de las órdenes de fabricación en Concreaceros S.A.S.

Objetivos específicos.

- Diseñar un algoritmo genético para la asignación de tareas en el área de máquinas en paralelo idénticas, teniendo como función objetivo minimizar el tiempo de terminación máximo.
- Plantear la heurística de Johnson para la asignación de tareas en el área de figuración estándar, logrando con ello reducir los tiempos ocios y de entrega.
- Formular las reglas de despacho para la planificación de las remisiones en cola, estimando una mejora en los criterios evaluativos.
- Validar el modelo propuesto: verificando su aplicabilidad y funcionalidad en el sistema de producción de Concreaceros SAS.

Metodología

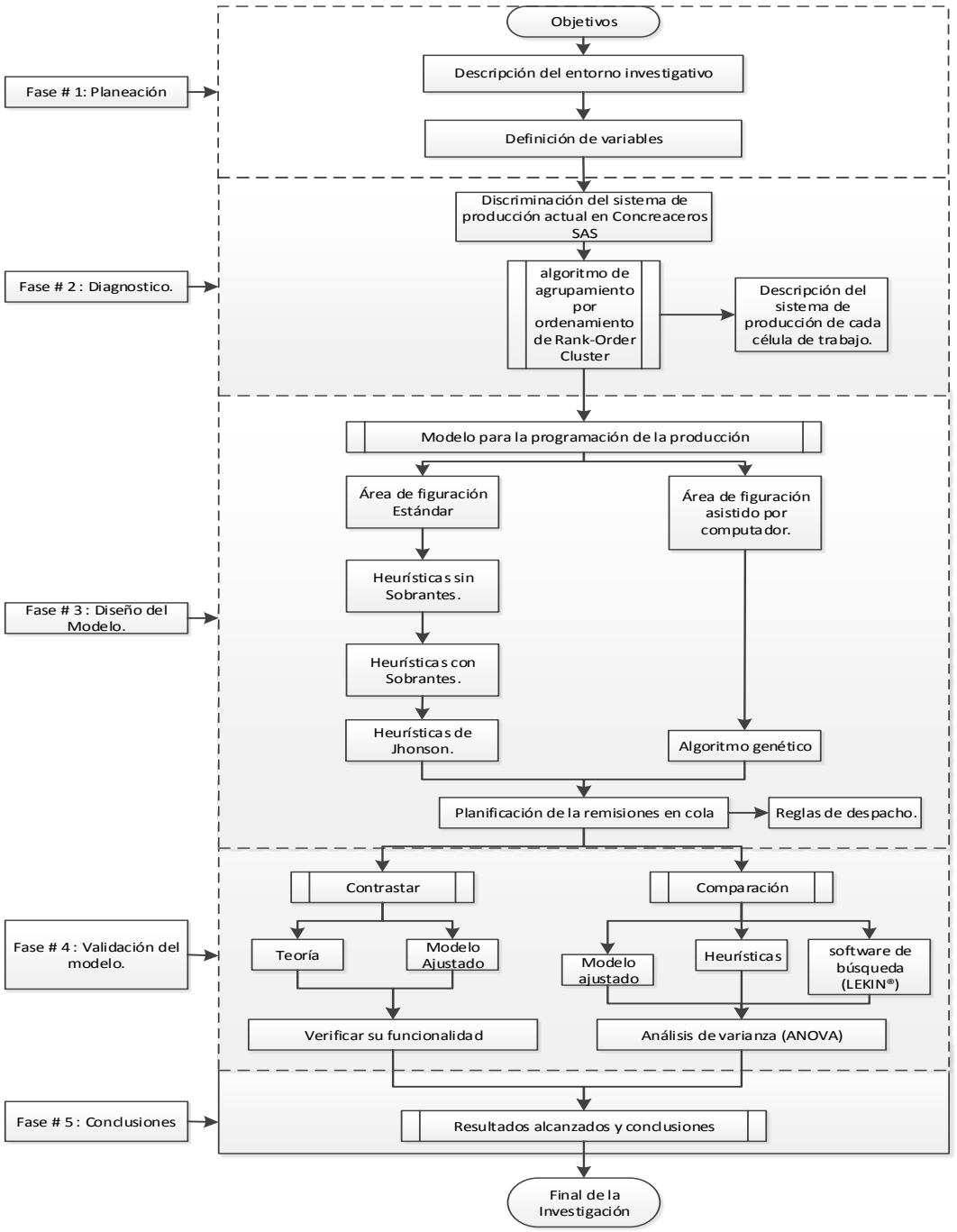


Figura 1. Diagrama de Fases de la metodología propuesta en esta investigación.

Fuente: Elaboración propia.

Para el presente proyecto de investigación se tuvo un estudio netamente cuantitativo, regido por diferentes fases discriminadas a continuación, aun así es pertinente resaltar que el tipo de investigación fue descriptivo, ya que; se definió variables imprescindibles que fueron ajustadas por medio de criterios evaluativos que describieron su comportamiento ante el sistema, todo esto fue enmarcado en fábrica de Concreaceros S.A.S. ubicada Sincelejo – Sucre, metodológicamente se ejecutó en cinco fases, como se puede apreciar en el diagrama de fases.

Cada una de las fases se encuentra relacionada y serán la fuente de información de entrada que necesitará la fase siguiente. Se inició con la planeación del proyecto como tal, enmarcando con ello el campo de investigación y planteamiento de objetivos que rigieron y dirigieron dicho proceso, luego cuando se tuvo definido el marco investigativo, se procedió a la siguiente fase, que fue un diagnostico exhaustivo al sistema productivo visto desde una perspectiva operacional, teniendo en cuenta como se encuentra distribuida las áreas de trabajo en la planta, su sistema de producción, asignación y programación de tareas y la planeación de las ordenes de los clientes (remisiones) en cola. En el proceso de reconocimiento e identificación de las células de trabajo en la fábrica y como se encuentran estas organizadas se utilizó el algoritmo de agrupamiento por ordenamiento de *Rank-Order Cluster* (King, 1982).

Luego de tener definido las células de trabajos, y conocer su proceso de asignación de tareas y sistema de producción, teniendo en cuenta variables y datos tomados, se procedió al desarrollo de un modelo ajustado a las necesidades suscitadas en el diagnóstico, que permita la inclusión de la investigación de las operaciones a este campo de estudio, por medio de meta heurísticas tales como: Algoritmo genético, para la asignación de tareas en un ambiente de máquinas en paralelo idénticas, y el diseño de heurísticas para las células de trabajos: que el ritmo de trabajo es impuesto por el operario y demanda un alto consumo en el tiempo de procesamiento de los lotes a fabricar, por ende se diseñó la heurísticas sin sobrantes y con sobrantes, para luego facilitar la implementación de la heurística de Johnson, y así proyectar unas disminución favorable en los tiempos ocio (muertos) trayendo consigo la disminución del tiempo de terminación máximo. Luego de haber programado las tareas de la remisión y teniendo estimado

su tiempo de terminación y fecha de entrega, se procede a planificar y organizar las remisiones en cola por medio de reglas despacho, diseñando las más comunes.

Para el desarrollo de las *Metaheuristicas* y *Heurísticas* mencionadas anteriormente, se utilizó como herramienta de programación: Matlab, se diseñó el modelo por medio de varios algoritmos en Matlab y así poder validarlo y con esto nos trasladamos a la fase siguiente, que consta de dos partes. La primera es verificar el funcionamiento de los algoritmos, comprobando con ello la correcta aplicación de la teoría a la práctica. Posterior a ello se procede a la segunda parte y es verificar y comparar por medio un análisis de varianzas (ANOVA), con respecto a los resultados arrojados en el modelo propuesto y un software de búsqueda (LEKIN® – Flexible Job-Shop Scheduling System) junto con heurísticas, para así validar su funcionalidad e idoneidad.

Y por último se considera el análisis de resultado y conclusiones en este proyecto.

Marco Teórico

A medida que pasan los años, no han cesado los esfuerzos investigativos en la programación de las operaciones (*scheduling*), debido a su complejidad y diversidad en los entornos. En lo que concierne a la administración de operaciones se han registrado a partir de la mitad del siglo XX, varias investigaciones sobre la asignación de tareas en los respectivos puestos de trabajos, gracias al deseo de poder responder de manare positiva a la satisfacción de los clientes, con respecto a los tiempos de entregas de sus productos (Velez Gallego , Castro Zuluaga, & Maya Toro, 2003). Para ello Morton y Pentico (1993) afirmaron que:

"Programar es el proceso de organizar, elegir y dar tiempos al uso de recursos para llevar a cabo todas las actividades necesarias, para producir las salidas deseadas en los tiempos deseados, satisfaciendo a la vez un gran número de restricciones de tiempo y relaciones entre las actividades y los recursos"

Indicándonos que la programación, consiste en la asignación de tareas bajo un patrón u orden en los respectivos puestos de trabajos, para ello, cada tarea requerirá de recursos, si son limitados incurrían problemas en la programación de la producción (Sipper & Bulfin, 1998), es decir, que estarán expuesto a tiempos improductivos en los puestos de trabajos y atrasos o adelantos en la fabricación de los lotes.

Pero, aun así, considerando la programación en los diversos ambientes de trabajos, se cuenta con un factor suspicaz, y es la complejidad relacionado con estos, ya que en su gran mayoría al aumentar el número de trabajos a programar o el número de puestos de trabajos y/o máquinas, crece desproporcionalmente la dificultad de encontrar la mejor de todas las soluciones (Vicentini & Puddu, 2003).

Por tal motivo las soluciones de este tipo de problemas se achantan en la definición de manera no determinística polinómial, y en su gran mayoría considerados *np-hard*, debido a su complejidad de búsqueda, ya que su espacio muestral es considerablemente grande (Garey & Johnson, 1979).

En los procesos de manufactura se encuentra precisada esta dificultad a nivel computacional, para obtener un nivel de incertidumbre acertado y confiable, haciendo referencia a la programación en estos ambientes de trabajos (Jop Shop – Scheduling). Cuando se cuenta con varias máquinas m = {M1, M2, …,Mm}, de las cuales pueden y deben realizar un conjunto de trabajos o tareas j = {1,2,…,n}, donde lo peculiar de estos sistemas productivos es la diversidad de operaciones que se deben hacer por tarea, es decir; la tarea j, debe pasar por las operaciones: O1 j, O2 j,…,Onj. Por tal motivo su campo muestral para buscar la secuencia acertada aumenta considerablemente gracias a la cantidad de máquinas y operaciones que requieran dicha tarea, sin olvidar el incremento desproporcional de las posibles soluciones al aumentar el número de tareas a programar, achantándose en problemas tipo NP – Hard (Cruz Chávez, Frausto Solís, & Juárez, 2009).

Incluso en sistemas productivos en los cuales la tarea debe pasar por un flujo de operaciones predefinidos y una secuencia que debe haber finalizado el proceso anterior para así poder pasar al siguiente, como se precisó anteriormente, pueden existir varias máquinas y varias tareas a programar, con la notable diferencia que las secuencias de las operaciones se encuentran definidas (f*low shop- Scheduling*), aun así, el problema subsiste, ya que cada trabajo tendrá un tiempo de terminación en cada máquina, teniendo consigo varios tiempos de terminación según el número de tareas a programar, llevándonos nuevamente a niveles computacionales complejos para su solución (MIRLEDY TORO, RESTREPO, & GRANADA, 2006)

La programación y sus dificultades inherentes al momento de asignar recursos limitados no solo se ve reflejadas en fábricas o talleres de producción, sino también en otros entornos, tales como en los puertos, donde pueden existir miles de formas para hacer un procedimiento, y al momento de ser evaluada cada alternativa o posible solución, bajo una función establecida; sea costo o tiempo, donde su resolución puede ser considerablemente difícil, hablando computacionalmente.

Casos notorios en donde se involucran variables entre tiempo y espacio, donde un buque al ser ubicado en un lugar determinado del muelle para atracar, donde se debe tener en cuenta que cada buque es una tarea a realizar y su operación puede ser cargue y/o descargue, el mismo

ocupa un espacio, algo limitado donde pueden atracar uno o varios a la vez, dependiendo del tamaño de los buques, y teniendo en cuenta que su tiempo de terminación se verá relacionado inversamente proporcional con el número de grúas asignadas: entre mayor grúas porticas asignadas menor tiempo tardara en realizarse la tarea, pero aun así se debe considerar que las grúas son limitadas y también se encuentran buques en cola, donde el tiempo muerto, de espera, tardanza y/o adelanto pueden ser penalizados, estos tipos de problemas son modelados bajo funciones objetivos que pretenden minimizar costos generales de penalizaciones al momento de atracar los buques y asignar grúas (Park & Hwan Kim, 2003).

El objetivo contextual de la programación de la producción, es poder cumplir con los tiempos de entregas de los lotes, por ende, un criterio evaluativo, que valoré a cada secuencia, permitiéndome con esto saber cuál es la acertada. Cuando se refiere a la entrega oportuna se utilizan por lo general funciones que evalúen el tiempo determinación de la secuencia pero aun así en algunos casos, al momento de programar y asignar una posible secuencia, el método de evaluación son regidos por costos, es decir: minimización de costos como función objetivo, por medio de restricciones que penalicen algunas condiciones, como lo es en el caso de la asignación de buques al muelle de ataque, aun así tiene estrecha relación con el tiempo de terminación, ya que entre mayor tiempo no deseado , mayor será el índice de penalización expresados en costos (Park & Hwan Kim, 2003).

Pero aun así en la mayoría de los problemas de programación y más en la gestión de la producción, al momento de modelarlas, se establecen en función del tiempo. El *Makespan* o tiempo de terminación máximo puede ser de vital importancia, al momento de evaluar las secuencias estimadas, ya que este nos dirá el tiempo en que en realidad tardará la secuencia en terminar, por tal motivo se obvia una función, bajo restricciones definidas, que pretendan buscar la secuencia que menor tiempo de terminación máximo consuma. (Hwan Kim & Park, 2014).

El *Makespan* es utilizado en muchos sistemas de producción, e incluso se han realizado investigaciones con fin de establecer secuencias que tiendan a disminuir el tiempo de terminación máximo en ambientes *Flow Shop*, (MIRLEDY TORO, RESTREPO, & GRANADA, 2006)**.** Partiendo en estos ambientes de producción, la implementación de heurísticas tales como

la de Johnson (1953), ayudaría a una disminución en el tiempo de terminación máximo (*makespan*), ya que dicha heurística trata de disminuir los tiempos muerto, trayendo consigo una disminución general en el tiempo de terminación máximo.

Guinet (1993) diseñó una heurística permitiendo tener como función objetivo minimizar el *Makespan* en máquinas en paralelo idénticas. No obstante la implementación de algoritmos genéticos (Holland, 1975), e incluso su adaptación para la solución de problemas referentes a la programación de la producción, se ven reflejados en los últimos años su aplicación en diferentes industrias, tales como investigaciones precedidas por Salazar y Medina (2013) concernientes al diseño de un modelo por medio de la utilización de algoritmos genéticos, con la finalidad de disminuir el tiempo de terminación máximo, constatando en ello la influencia de manera positiva con respecto a los tiempos de entrega de los lotes a fabricar, trayendo consigo la satisfacción de los clientes en ese aspecto, bajo un ambiente de máquinas en paralela con tiempo de preparación dependientes (Salazar - Hornig & Medica -S, 2013).

Además algunas investigaciones registradas en la industria textil con la implementación de algoritmos genéticos para la realización de la programación de la producción de una manera eficiente, ejemplo de ello la investigación realizada por Juan Lopez vargas (2013) en ambientes flow Shop híbrido flexible (López Vargas, 2013) y en líneas de flujos sencilla bajo este mismo patrón, resaltando los buenos resultados arrojado por medio de esta meta heurística y los tiempos razonables de ejecución a nivel computacional (Lopez, Giraldo, & Arango, 2015), e incluso se registran investigación en la asignación de recursos en talleres de pedidos mecanizados con la utilización de algoritmos genéticos bajo maquinas en paralelo (Pérez Pérez, Pérez Castillo, & Jiménez Bahri, 2014), indicando la aceptabilidad de tomar el tiempo de terminación máximo, como criterio evaluativo acertado, con respecto a la programación y asignación de recursos.

Pero considerando los problemas relacionados a la asignación de recursos (tareas) en una sola maquina (*single machine*), su complejidad es un poco moderada, ya que gracias a su solución por medio polinomiales (Montoya Torres, Paternina Arboleda, & Frein, 2002), pero aun así se resalta la aplicación de métodos exactos y heurísticas que pueden ayudar al rendimiento del sistema, ya que se puede establecer criterios evaluativos siempre y cuando se conozca el tiempo de

procesado de la tarea y la fecha de entrega de la misma, se puede contractar con los siguientes criterio: número de tareas que se entregas a tiempo, numero de tareas que se entregaran tarde, tardanza máxima, tardanza promedio, adelanto máximo, adelanto promedio y el tiempo flujo, variables que ayudaran a la planificación de la mismas, siempre y cuando su evaluación de haga de manera preliminar para ver con cual heurística o método exacto logra un mejor nivel de satisfacción con respecto a los tiempos de entregas, es pertinente resaltar que en estos casos solo nos puede servir de guía a nivel general el tiempo de terminación máximo (*Makespan*) ya que en estos ambientes de producción el orden de los sumandos no altera el resultado (CHASE, JACOBS, & AQUINALO, 2009).

Marco Conceptual

Reseña de la empresa de estudio

Nombre.

Concreaceros S.A.S.

NIT.

900507746

Logotipo.

Figura 2. Logotipo de la empresa tomada para la realización de la investigación.

Misión.

En CONCREACEROS S.A.S comercializamos el acero y servicios de losas pos tensadas para la industria y la construcción, e innovamos con tecnología de punta para brindarles a nuestros clientes productos más competitivos en sus inversiones. CONCREACEROS S.A.S mantiene alianzas estratégicas con Carlos José Palencia Diego que es una organización dedicada a ofrecer servicios de diseño estructural y construcción de obras civiles en el sector público y privado de la costa atlántica.

Visión.

En el 2020 CONCREACEROS S.A.S se perfila como una empresa líder en la costa atlántica, con excelentes resultados financieros y en crecimiento constante en atención, optimización de

sus procesos y cubrimiento regional. Distinguiéndose por la permanente incursión en productos y servicios con tecnología de punta.

Diagnóstico del sistema de producción actual en Concreaceros SAS.

Al tomar la empresa las especificaciones y requerimientos de los clientes; pactan de mutuo acuerdo una fecha de entrega. Posterior a eso se procesa la información que será remitida en la planta de producción, y así se programa para su elaboración. Cabe resaltar que actualmente el encargado de la producción, ordena las remisiones con respecto al tiempo de entrega, colocando primero la remisión con fecha de entrega próxima; a eso se le suman los pesos o importancia, factor que puede contribuir a la organización de las mismas. Al momento de escoger la remisión a procesar, se mira como primer punto lo que se va a realizar, teniendo en cuenta el tipo de acero a utilizar, ya que la remisión a elaborar en la planta consta de variados insumos, para la figuración se utilizan diversos aceros que varían según su grosor. En la siguiente tabla se muestra el tipo de acero usado en la empresa con su respectivo calibre.

Tabla 1

Tipos de aceros utilizados en la empresa.

Tipo de acero	Calibre (pulgadas)	Calibre (milímetros)
#2	1/4	6,35
#3	3/8	9,525
#4	1/2	12,7
#5	5/8	15,875
#6	3/4	19,05
#7	7/8	22,225
#8	1	25,4

Nota: Actualmente la empresa cuenta con siete tipos de aceros, que utiliza como insumo en sus procesos estándar.

Cuando se van a asignar las diferentes tareas en los puestos de trabajos, el patrón de secuencia es convencional, y en algunos casos se hace de manera arbitraria. La remisión se encuentra clasificada según el tipo de acero, organizada de menor a mayor según el calibre a utilizar. En cada sección el tipo de acero, posee las respectivas figuras que hace alusión a los productos finales deseados, con sus correspondientes medidas y las cantidades a fabricar. Dichas remisiones solo deben ser manejadas por el encargado de la producción en la planta, esa misma persona es la responsable de distribuir la información. El medio de divulgación a los respectivos puestos de trabajo se hace por medio de etiquetas, las cuales poseen las características del material a utilizar, como longitud total que toma la figura, figurado y cantidad.

Se hace una excepción con los aceros tipo número tres, ya que este viene por chipas (rollos), por ende, las tareas que requieran de este tipo de acero, son programas en el área de figuración asistidos por computadores. Actualmente se cuenta con dos máquinas. El figurado se puede hacer en cualquiera de las dos, por lo que los trabajos pueden realizarse en una o ambas máquinas al ponerlas a trabajar en paralelo, su ritmo es igual, ya que cuentan con la misma capacidad de producción. Las cantidades exigidas en esta sección son grandes en comparación con los demás tipos de acero, por lo que la producción en estas máquinas es asistida por computadoras, teniendo así un proceso automatizado.

El resto de acero utilizado se encuentran en varillas estándar es decir en longitudes rectas de seis, nueve y doce metros; en la remisión se aclara el tipo de varilla y el figurado a realizar, para esto se cuenta con el área de figuración estándar, que consta del proceso de cortado y doblado. Actualmente se tienen dos máquinas de cortado y tres de doblado. El proceso a seguir en esta área es considerado como producción intermitente en dos máquinas ya que tiene cuatro posibles diferentes formas de procesar las tareas, ya sea que solo requiera cortar, doblar, cortar y luego doblar o viceversa. Pero en el tiempo de observación se detectó que aproximadamente el ochenta por ciento de las tareas asignadas en el área de figuración estándar se deben cortar y luego doblar, prospectivamente, se aclara que no es recomendable el doblar y luego cortar, ya que el encargado de la distribución de las etiquetas se encuentra en el área de cortado, además por facilidad de manejo debido al espacio, en consecuencia, es acertado definir ese flujo para el

procesado del acero en esos casos. El ritmo y forma de elaborar los trabajos en esta área son impuestos y dirigidos por los operarios del área de figuración estándar.

En la planta se cuenta con otra área de producción y es el área de grafilado, cuenta con dos máquinas, una es la trefiladora, encargada de procesar el alambrón y bajar su diámetro, para luego ser pasado por la máquina de enderezado y cortado, para así obtener varillas milimetradas. Aun así, se fabrica para mantener un inventario, lo que nos deja claro que el proceso es diferentes a los anteriores, ya que en los mencionados anteriormente, la producción inicia cuando se tiene claro el pedido del cliente, pero aquí es todo lo contrario, se fabrica por temporadas, ya que se pueden fabricar diferentes tipos de varillas milimetradas, que varían según el grosor, que son almacenadas y luego vendidas, por lo tanto no vienen encargadas ni especificadas en las remisiones que bajan a la planta de producción.

Para el adecuado estudio en el área de producción, se hizo pertinente la clasificación y agrupación de los respectivos puestos de trabajos en células de trabajos. “Cada célula se diseña para producir una variedad limitada de las configuraciones de parte; es decir, la célula se especializa en la producción de un conjunto determinado de partes similares, según los principios de la tecnología de grupos” (Groover, 1997, pág. 23). Lo que nos deja claro precisar, que en base a los insumos y en su proceso, la naturaleza del producto debe ser similar para su agrupamiento. Para la distribución se utilizó el algoritmo de agrupamiento por ordenamiento de Rank-Order Cluster (King, 1982), para ello se debió precisar una matriz de incidencia, como se muestra en la siguiente tabla 2.

Tabla 2

Algoritmo de agrupamiento por ordenamiento de Rank-Order Cluster

Evaluador	Insumo	Maquinas						Salida	Peso
		Cortadora	Dobladora	Estribadora 1	Estribadora 2	Trefiladora	Enderezadora		
2	acero tipo #2	1	1					Acero figurado estándar	6
4	acero tipo #4	1	1					Acero figurado estándar	6
8	acero tipo #5	1	1					Acero figurado estándar	6
16	acero tipo #6	1	1					Acero figurado estándar	6
32	acero tipo #7	1	1					Acero figurado estándar	6
64	acero tipo #8	1	1					Acero figurado estándar	6
128	acero tipo #3			1	1			Acero figurado	24
256	alambrón					1	1	varillas milimetradas	96
	Peso	126	126	128	128	256	256		

Nota: Para la distribución se utilizó el algoritmo de agrupamiento por ordenamiento de Rank-Order Cluster (King, 1982), para la identificación de las células de trabajo.

Como se puede observar, se encuentran actualmente en la planta tres células de trabajos, las cuales mencionamos anteriormente como: el área de figuración estándar, figuración asistida por computador y grafilado.

El tema a considerar es la mano de obra asignada por cada célula de trabajo, que se refleja en la siguiente tabla.

Tabla 3

Asignación de la mano de obra.

Célula de trabajo	Maquinaria	Número de operarios	Número total de operarios por célula de trabajos
1	Estribadora 1	2	4
	Estribadora 2	2	
2	Cortadora	3	6
	Dobladora	3	
3	Trefiladora	1	2
	Enderezadora + cortadora	1	

Nota: El número de operarios puede aumentar en algunas ocasiones, pero por lo general se maneja esta cantidad de operarios.

En promedio se tiene un total de doce operarios asignados en el área de producción de Concreaceros SAS. Para resumir lo dicho en esta reseña se añade a continuación una tabla, para ver la naturaleza de cada célula de trabajo seguido de su sistema de producción.

Tabla 4

Células de trabajos establecidas.

Célula de trabajo	Insumo	Maquinaria	Proceso	Salida	Tipo de producción
1	Acero tipo #3 Acero tipo #3	Estribadora 1 Estribadora 2	Figurado asistido por computadora	Acero figurado	Single machine Maquinas en paralelo
2	Acero tipo #2 Acero tipo #4 Acero tipo #5 Acero tipo #6 Acero tipo #7 Acero tipo #8	Cortadora Dobladora	Figurado del acero estándar	Acero figurado	Producción intermitente
3	Alambrón rollo grafilado	Trefiladora Enderezadora + cortadora	Trefilado	Varillas milimetradas	Producción continua

Nota: La célula de trabajo 1 tiene la libertad de programar sus actividades en una sola maquina o ponerlas a trabajar en paralelo.

Falencias encontradas en el sistema actual de producción.

Después de haberse dado una pequeña reseña junto con el reconocimiento actual de la planta, tras el estudio de este, se detectaron algunas falencias en las diferentes células de trabajo. Todo apunta a la errónea programación y control de la producción que se ve reflejada en el parcial cumplimiento en las fechas de entrega de las remisiones. Es preciso anotar que el estudio sólo se realizó en el área de producción para simplificar y conceptualizar la investigación, dejando a un lado áreas tales como la administrativa y logística. Para una mayor apreciación de la problemática se hará el estudio por cada célula de trabajo, para luego hacer mención de los

aspectos generales del sistema actual de programación y planificación de la producción en Concreaceros S.A.S.

Áreas de figurado asistido por computador.

El área de figuración asistido por computadora, cuenta, como ya se dijo, con dos máquinas, la referencia de la primera es Alba y la segunda es Mep; la diferencia radica en que la primera es más antigua que la segunda, pero aun considerando esto, su proceso de producción es el mismo ya que cada una de las máquinas cuentan con dos devanadores, indicando con esto que pueden trabajar con doble hilo, es decir; la maquina fabrica dos unidades a la vez. La falencia encontrada en este aspecto es que la máquina Alba en su gran mayoría trabaja con un solo hilo, desperdiciando así la mitad de su capacidad operativa, dando con esto una sub utilización del equipo del 50%.

Otro punto a considerar, es que los productos realizados es esta célula de trabajo varían de manera considerable su figurado, es decir; su figura y sus longitudes, teniendo con eso diversidad en los tiempos ciclo de cada lote de producción. El desconocimiento del tiempo que tarda en salir una unidad hace que en parte no se estime a ciencia cierta el tiempo que tardaría el lote en realizarse, aumentando con esto el descontrol en la producción, y una errada planificación de los puestos de trabajos. No obstante, en algunos casos los tiempos son estimados, lográndose en ocasiones tener noción sobre que lote tarda más en producirse en comparación con otro, y partiendo de aquí me gustaría resaltar otra falencia detectada y es el pleno desconocimiento del tiempo de terminación máximo (*makespan*). El *Makespan* nos indica el tiempo que se tomaría en producirse todos los trabajos asignados en esta célula y que conciernen a la remisión en proceso. Para mejor apreciación se explicará en el siguiente ejemplo en este mismo ambiente de trabajo:

Tabla 5

Ejemplo ilustrativo.

Tarea	Cantidad	Tiempo de Procesamientos del lote
A	10	45
B	5	30
C	30	60
D	50	100
E	40	90
F	30	64
G	10	20

Nota: Ejemplo ilustrativo, cada tarea contiene el número de unidades a producir y en tiempo que tardaría en realizar en unidades de tiempo.

Del ejemplo se puede apreciar siete trabajos que conciernen a una remisión por fabricar. Se asocia que las siete tareas son aceros figurado que requieren de insumo aceros número tres , es decir; que su proceso se llevará a cabo en el área de figurado asistido por computadoras, como dije anteriormente, se cuenta con dos máquinas y cada tarea lleva la cantidad a producir y el tiempo que se tarda en fabricar todas esas unidades, por lo tanto se podrá asignar en cualquiera de las dos máquinas, y para ellos se precisa que para el primer trabajo programado en la maquina uno y dos su tiempo de inicio es cero ya que las máquinas se encuentran disponibles y los trabajos no se pueden interrumpir para poder apreciar bien el proceso. Se procede con el cálculo del tiempo de terminación máximo (*Makespan*).

Tabla 6

Trabajos asignados en la maquina 1.

	Tarea	Cantidad	Tiempo de Procesamientos del lote	Tiempo de inicio	Tiempo de terminación
Maquina 1	A	10	45	0	45
	B	5	30	45	75
	C	30	60	75	135

Nota: El tiempo inicio de la tarea asignada de primero es cero, ya que la maquina se encuentra disponible, el resto del tiempo depende del tiempo transcurrido hasta el momento.

Tabla 7

Trabajos asignados en la maquina 2.

	Tarea	Cantidad	Tiempo de Procesamientos del lote	Tiempo de inicio	Tiempo de terminación
Maquina 2	D	50	100	0	100
	E	40	90	100	190
	F	30	64	190	254
	G	10	20	254	274

Notal: El tiempo inicio de la tarea asignada de primero es cero, ya que la maquina se encuentra disponible, el resto del tiempo depende del tiempo transcurrido hasta el momento.

Cómo se puede observar en la tabla, tres de los siete trabajos fueron asignados en la máquina número 1 y el resto de los trabajos fueron asignados en la máquina número 2, con sus concernientes tiempos de producción. En la parte izquierda de la tabla se anexan dos columnas las cuales, la primera se llama tiempo de inicio que corresponde al comienzo de las tareas, bajo

el horizonte de las unidades de tiempo; y la segunda, tiempo de terminación, qué hace referencia a la culminación de la actividad bajo el horizonte de las unidades de tiempo. Como es continuo, el tiempo de terminación de la tarea anterior es el tiempo de inicio de la tarea siguiente; dicho proceso se hace de manera acumulativa, indicándonos que en el inicio de la tarea asignada en el puesto número tres sería la suma de los dos tiempos de producción de las tareas asignadas en el puesto uno y dos. Para mayor facilidad se visualiza en un gráfico de Gantt.

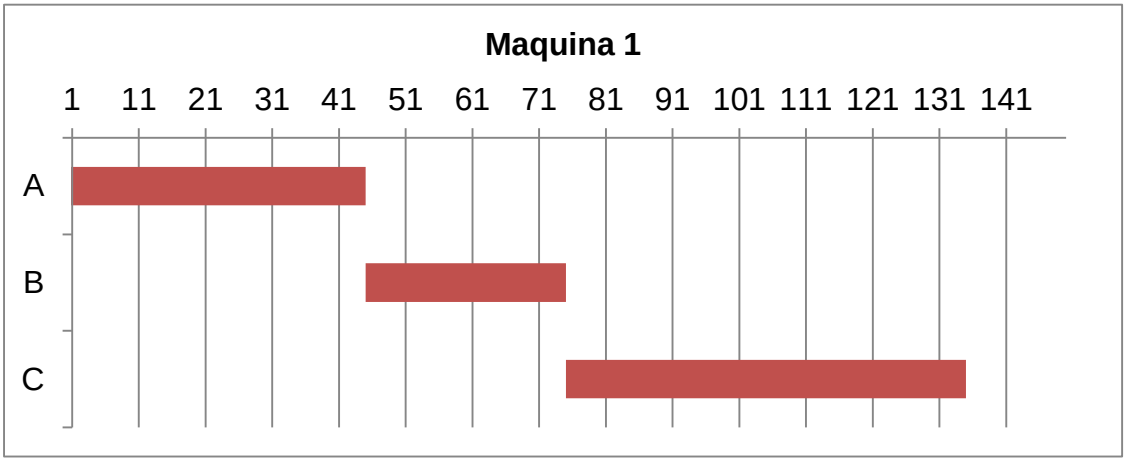


Figura 3: grafico de Gantt, elabora en Excel 2013, se presenta la secuencia asignada en la maquina 1 bajo un horizonte de tiempo.

Fuente: Elaboración propia.

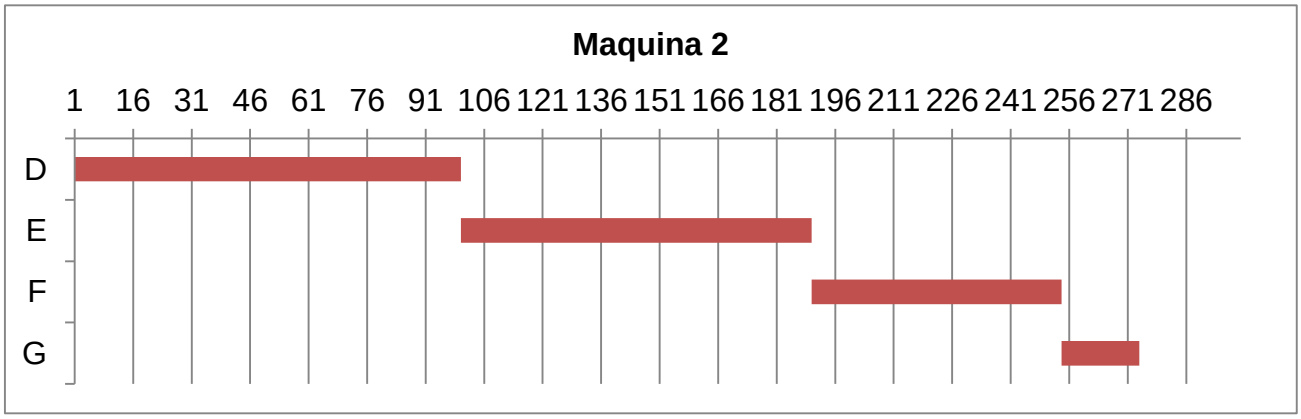


Figura 4: grafico de Gantt, elabora en Excel 2013, se presenta la secuencia asignada en la maquina 2 bajo un horizonte de tiempo.

Fuente: Elaboración propia.

Como se puede observar el tiempo de terminación máximo de las últimas tareas asignadas en las dos máquinas es 135 y 274 unidades de tiempo, por lo tanto, se toma el máximo entre ellos, indicándonos que el tiempo que tardó en producir las siete tareas en dicho ambiente fueron 274 unidades de tiempos; con un tiempo ocio de 139 unidades tiempo para la maquina 1. Dicha tarea o actividad no es puesta a evaluación, al momento de programar la producción en Concreaceros S.A.S, algo negativo, ya que desestima el tiempo que en realidad van a tardar realizando las respectivas tareas. Además, me gustaría resaltar otra falencia y es el desprecio también en la importancia del orden en que son asignadas las tareas en las máquinas en paralelo, ya que puede variar el tiempo de terminación de la remisión, para ello se adoptará que en la maquina 1 son asignados los trabajos E, F, B y G y en las máquinas dos los trabajos C, A y D del ejemplo anterior. Realizando los cálculos necesarios para hallar el tiempo de terminación de la remisión (*Makespan*) y su tiempo ocio, se tiene lo siguiente.

Tabla 8

Trabajos asignados en la maquina 1.

	Tarea	Cantidad	Tiempo de Procesamientos del lote	Tiempo de inicio	Tiempo de terminación
Maquina 1	E	40	90	0	90
	F	30	64	90	154
	B	5	30	154	184
	G	10	20	184	204

Tabla 9

Trabajos asignados en la maquina 2

	Tarea	Cantidad	Tiempo de Procesamientos del lote	Tiempo de inicio	Tiempo de terminación
Maquina 2	C	30	60	0	60
	A	10	45	60	105
	D	50	100	105	205

Nota: El tiempo de terminación máximo se encuentra en esta máquina, por lo tanto, el tiempo que tardo en realizase las tareas fue de 205 unidades de tiempo, mejor a la anterior secuencia.

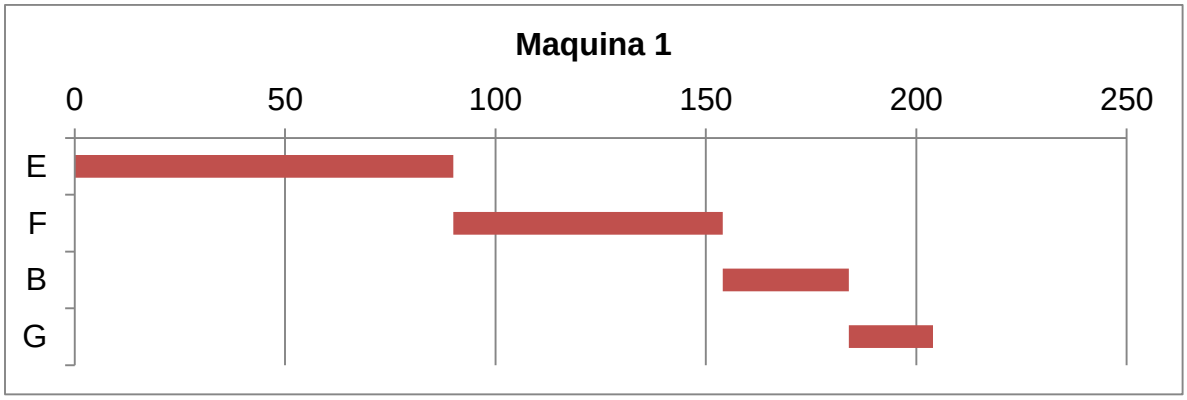


Figura 5: grafico de Gantt, elabora en Excel 2013, se presenta la secuencia asignada en la maquina 1 bajo un horizonte de tiempo.

Fuente: Elaboración propia

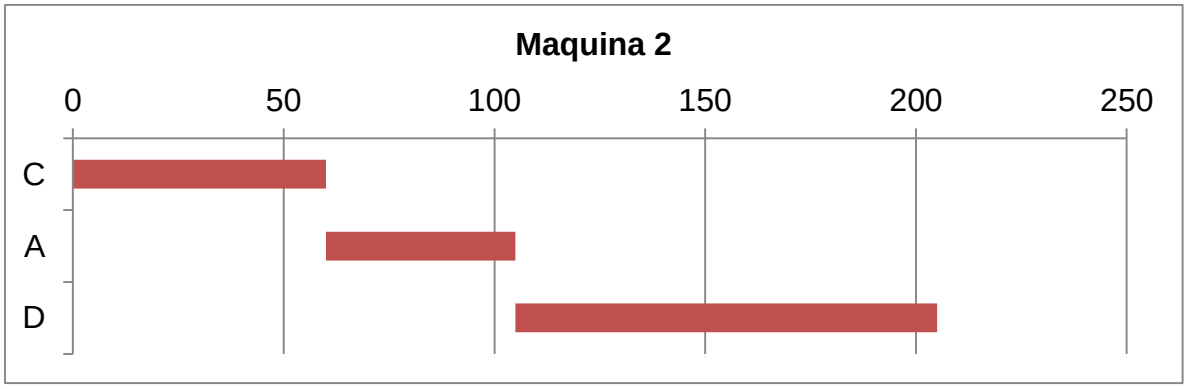


Figura 6: Grafico de Gantt, elabora en Excel 2013, se presenta la secuencia asignada en la maquina 2 bajo un horizonte de tiempo.

Fuente: Elaboración propia.

Se puede notar que el tiempo de terminación en este caso fue de 205 unidades de tiempos. Comparado con la secuencia anterior, podemos notar que disminuyó 69 unidades de tiempo; y su tiempo ocio solo fue de una unidad de tiempo. Lo que deja claro que esta última secuencia fue mejor en comparación a la anterior.

A esto se le suma la dificultad de solucionar este tipo de problemas, ya que para poder evaluar todas las posibles combinaciones (secuencias) que se generarían a partir del número de tareas a

programar, se tendría $(n!)^m$,donde n es el número de trabajos y m el número de máquinas disponibles para asignar los respectivos trabajos, solamente con 7 trabajos y dos máquinas disponibles para programar se tendrían 25.401.600 secuencias diferentes, haciendo nula su exploración por medio de métodos convencionales o manuales abriendo paso a métodos computacionales que permitan la exploración de una parte considerable de las secuencias posibles.

Áreas de figurado estándar.

Como se mencionó anteriormente, esta célula de trabajo cuenta con dos procesos: cortado y doblado; también se dijo que el sistema de producción es intermitente, pero aproximadamente el ochenta por ciento de las tareas asignadas requieren de cortar y luego doblar. Ante este flujo se vio que la actividad más lenta es cortado, por ende dicha actividad restringe el proceso de figurado, ya que en su gran mayoría, los trabajos asignados en esta área, requieren de esta operación, y su proceso tarda un tiempo considerable, dándose en la mayoría de los casos privación en el proceso, es decir; que la actividad siguiente debe detenerse porque no hay trabajos disponibles para ella (CHASE, JACOBS, & AQUINALO, 2009), aumentando con ello el tiempo improductivo de los operarios del área de doblado.

En esta área de trabajo persisten las falencias con respecto a las secuencias, sin su previa evaluación, excluyendo criterios como el tiempo de terminación máximo; gracias a que en su gran mayoría no se tenga conocimiento del tiempo que tarda un lote en fabricarse, a esto se le suma las dificultades presentadas al momento de suministrar los insumos requeridos para la elaboración de las figuraciones estándar, ya que para este proceso, se requiere, en algunos casos varios insumos, por lo tanto el desconocimiento de la cantidad de ellos, que van a utilizarse en ese instante para la culminación del proceso; trae como resultado una demora en la asignación de los insumos, debido a que el puente grúa, qué es el encargado de suministrar los rollos de varillas, se encuentra gran parte del tiempo ocupado en el despacho de órdenes.

Aspectos generales de la programación de remisiones.

Los problemas resaltados anteriormente se hacen evidentes en el desarrollo de la remisión, pero visto desde una perspectiva general se encuentran falencias al momento de planificar y establecer el orden de producción de las remisiones. Estas, poseen una fecha de entrega, por lo que es necesario hacer un estimativo del tiempo que tardaría en elaborarse. Pero desafortunadamente el desconocimiento, previo, de algunos criterios evaluativos tales como:

- Número de remisiones que no se entregarán a tiempo.
- Número de remisiones realizadas con anterioridad.
- Tardanza máxima y promedio del total de remisiones a evaluar.
- Adelantó máximo y promedio del total de remisiones a evaluar.
- Tiempo total utilizado para la realización las remisiones a evaluar.
- Carencia de una función objetivo que evalúe la secuencia estimada.

Otro aspecto a considerar en la recepción de órdenes y la liberación de las mismas, ya que el desajuste y la no implementación de políticas adecuadas pueden generar desórdenes y atrasos como se aprecian actualmente en la empresa.

Organización y diseño de los algoritmos del modelo propuesto

Lo que esperamos con este modelo es una disminución favorable en los tiempos de entrega de los lotes de producción, por medio de un diseño ajustado a las necesidades y restricciones de las respectivas células de trabajo, con la finalidad de programar y controlar la producción en Concreaceros SAS.

Área de figurado asistido por computadora

Lo peculiar de esta célula de trabajo, es la variedad de trabajo que puede realizar, cambiando su longitud y figurado; trayendo consigo finitos tipos de producto con diferentes tiempos ciclos en su gran mayoría. Para ello se hizo un estudio de tiempo en esta área de trabajo, en los cuales

primero se clasificaron y se agruparon en familia, los productos similares, ya que, en algunos, su figurado es más complejo y la maquina tarda más en realizarlos. La clasificación según su forma y longitud es la siguiente:

Tabla 10

Clasificación según su figura.

Según su figurado.
Gancho
Estribo
Varillas rectas
Figurado tipo L
Otras figuras.

Nota: Clasificación adoptaba para facilidad del estudio.

Tabla 11

Clasificación según su longitud.

Especificación	Rango de medida en metros
Pequeño	X > 0 & X < =1
Mediano	X > 1 & X <= 2
Grande	X > 2

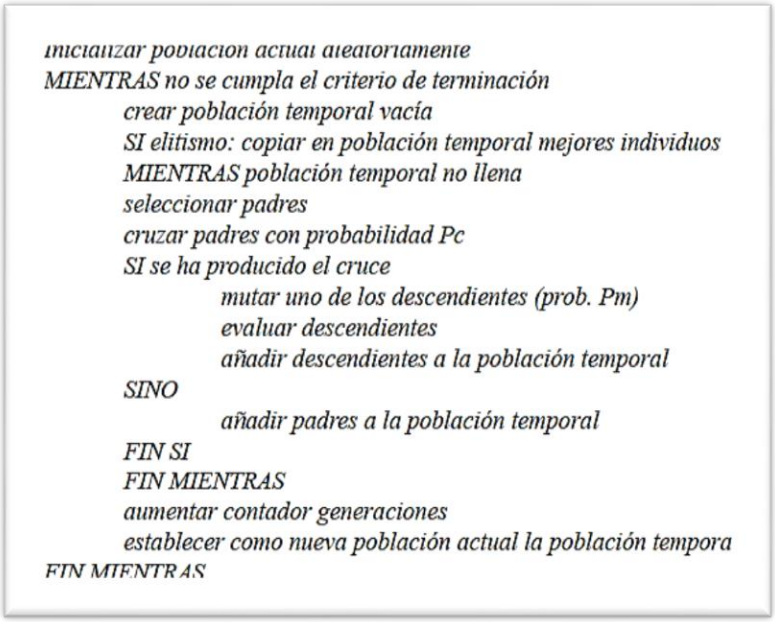
Nota: Clasificación adoptaba para facilidad del estudio.

Debido a lo corto del estudio, no se logró recolectar el tiempo-ciclo de todos los productos en esta área; pero aun así se tomó los que tienen mayor incidencia. Al momento de desarrollar la metodología planteada para esta área, se hace indispensable el conocimiento del tiempo de producción, para ello se definió si por algún motivo el producto no se encuentra exactamente

registrado en la muestra tomada, se tomarán aquellos que estén afines con su figurado y se aproximen en su longitud, para así poder estimar el lapso de tiempo que tardaría en fabricarse la respectiva referencia. Además, se define como ambiente de trabajo para esta célula, un sistema de máquinas en paralelo, idénticas, y para la maquina ALBA se obvia que su eficiencia con respecto a su capacidad es del cien por ciento, es decir; se trabajará doble hilo, (se realiza dos unidades a la vez) al igual que la maquina MEP.

Desarrollo del modelo.

Como se pudo denotar anteriormente, lo que se pretende es establecer un mecanismo para escoger una secuencia acertada para la asignación de tareas en comparación a otras, bajo una función objetivo, para este caso la utilizada será la minimización del tiempo de terminación máximo de las tareas asignadas, es decir; que se evaluará cada secuencia creada según su tiempo de terminación para decir que tan acertada es, y así escoger la mejor (la secuencia que consuma menos tiempo en realizarse). Para ello se utilizó como herramienta un algoritmo de búsqueda y optimización, conocido como algoritmo genético, y para ello se tuvo que definir algunas restricciones para su correcta aplicación en este ambiente de trabajo. Además, se hace indispensable la ayuda de un programa que permita el ingreso de las restricciones y especificaciones del algoritmo genético modificado, para ellos se utilizó Matlab.



```
Inicializar población actual aleatoriamente
MIENTRAS no se cumpla el criterio de terminación
        crear población temporal vacía
        SI elitismo: copiar en población temporal mejores individuos
        MIENTRAS población temporal no llena
        seleccionar padres
        cruzar padres con probabilidad Pc
        SI se ha producido el cruce
                mutar uno de los descendientes (prob. Pm)
                evaluar descendientes
                añadir descendientes a la población temporal
        SINO
                añadir padres a la población temporal
        FIN SI
        FIN MIENTRAS
        aumentar contador generaciones
        establecer como nueva población actual la población tempora
FIN MIENTRAS
```

Figura 7: Pseudocódigo que muestras el funcionamiento de un algoritmo genético.

Fuente: (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010, pág. 15).

Como se puede apreciar del Pseudocódigo, el primer paso es generar una población, un número determinados de individuos, en este caso un individuo hace alusión a una secuencia, una posible solución, el individuo está conformado por alelos, y cada alelo es la respectiva tarea asignada en un puesto determinado de manera aleatoria. Al momento de crear la población se debe hacer lo más aleatorio posible para así poder explorar mejor el espacio muestral (Holland, 1975), como se mencionó anteriormente, el orden importa, ya que al ser alterado el resultado cambia en su gran mayoría, por ende, cada individuo posee un orden o una secuencia conocida en este campo como cromosoma, para una mayor apreciación se mostrará la siguiente figura.

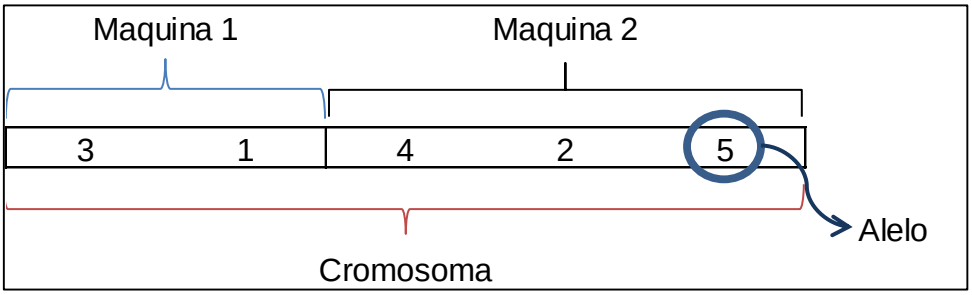


Figura 8: Individuo creado a partir de cinco trabajos asignados en dos máquinas en paralelo, mostrando las partes que lo conforman.

Fuente: Elaboración propia.

Posterior a la creación de los individuos, se procede a la evaluación de manera particular, para ello se contará con una función objetivo que dirá que tan sobresaliente es el individuo en el entorno donde se encuentra. Para ello se ha establecido la minimización del *Makespan*, cada secuencia llevará su tiempo de terminación. En la evaluación puede que algunos sean buenos y otros no tanto, pero lo que se querrá es darles una mayor oportunidad a aquellos individuos más aptos, y así poder someterlos a los operadores genéticos definidos a continuación, con la finalidad de mejorar su rendimiento por medio de la extracción del gen que hace referencia a la esencia del individuo, es decir, la combinación interna que contribuya a su idoneidad y así transmitirlo a sus descendientes (mejorar la secuencia).

Cabe resaltar que el número de veces que se pondrán en marcha los operadores genéticos (iteraciones) será definido por el programador y es independiente al número de individuos creados en la población, ya que este también será especificado por el programador antes de pasar a los operadores genéticos, cuyos componentes son: selección, cruce y mutación. Cuando se tenga consigo un número considerablemente de secuencias con su tocante evaluación, se colocará en marcha el operador de selección por torneo de forma determinísticas, ya que se tomarán dos individuos al azar y se escogerá según la función objetivo, el mejor de ellos en nuestro caso es el que tenga el menor tiempo de terminación (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010). Como este algoritmo trata de simular lo que se aprecia en las especies, el más apto tiene mayor posibilidad de poder aparearse y así obtener descendencia debido a su buena condición física; para ello debe contar con otro individuo que será su pareja (Holland, 1975). Lo

mismo sucederá en este caso, el proceso de selección se hará dos veces, para así obtener dos posibles padres a considerar.

El operador siguiente es el de cruzamiento. Es preciso anotar que la probabilidad que ocurra debe ser elevada, que tienda al cien por ciento, ya que los individuos se encuentran en un nivel alto de idoneidad para procrear. Para este caso el programador tendrá la libertad de escoger la probabilidad de cruzamiento. Se recomienda utilizar una del noventa y ocho por ciento. Después de haber escogido los dos posibles padres, se generará al azar un numero aleatorio, si se encuentra en el rango probabilístico de cruzamiento, se realizará el operador, en caso contrario se pasa a la iteración siguiente. Al momento de generar el cruzamiento se debe tener en cuenta que se generarán dos hijos, y para ellos se utilizará un cruzamiento en dos puntos y posterior al cruzamiento se verifican en los hijos creados, que alelo se encuentran repetidos y así cambiarlos de manera aleatoria por el alelo faltante, como se ve en la siguiente figura.

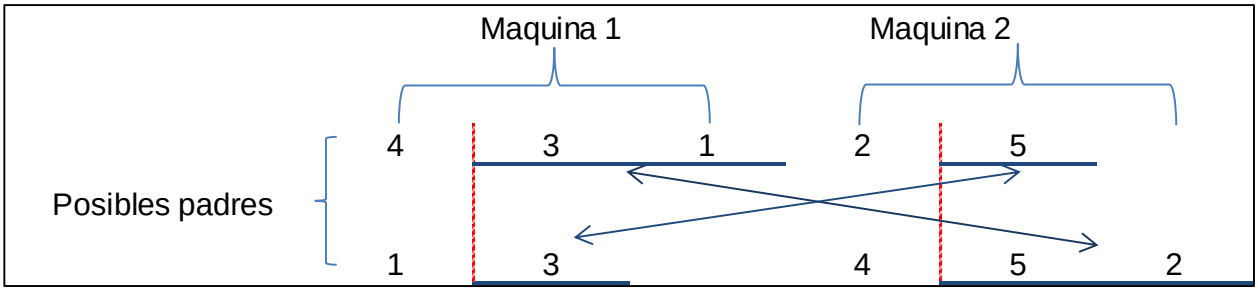


Figura 9: Representa la metodología de cruzamiento, los que se encuentran antes de línea intermitente roja quedan fijos, mientras los que están después son intercambiados.

Fuente: Elaboración propia.

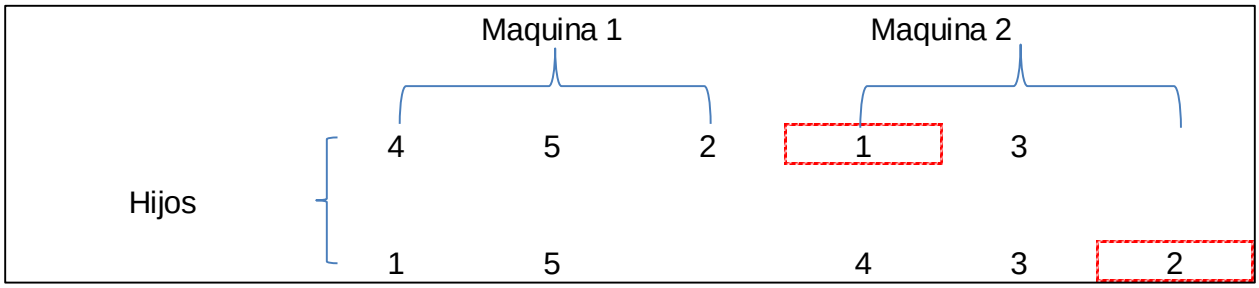


Figura 10: Hijos creados a partir de los padres mencionados en la figura 9, los que están encerrados en las líneas intermitentes rojas, son aquellos que fueron cambiados por los alelos faltantes ya que se encontraban repetidos.

Fuente: Elaboración propia.

Después se evalúa a los hijos creados con respecto al tiempo de terminación máximo, para escoger al mejor de los hijos, el que tenga menor de tiempo de terminación y así pasar al operador siguiente. Luego se toma al mejor hijo y se pasa al siguiente operador.

La mutación es una alteración genética que por lo general se desarrolla bajo una probabilidad muy baja, pero aun así está presente en este algoritmo de búsqueda. Para ello el programador tendrá la libertad de asignar la probabilidad de mutación, se recomienda una menor o igual al uno por ciento, si al generar el número aleatorio se encuentra en ese rango probabilístico de mutación, se realizará el operador, en caso contrario se tomará al hijo sin mutar y se ingresará en una población nueva, conformada por individuos de la siguiente generación; y si cae en un rango permisible para mutar, se elegirán al azar las posiciones del alelo a cambiar en las dos máquinas. Posterior a ello, se evaluará si el hijo mutado mejoró su tiempo de terminación, y si resulta afirmativo se ingresará el hijo mutado en la población nueva, en caso contrario se ingresará el hijo sin mutar, así como se nota en la siguiente figura.

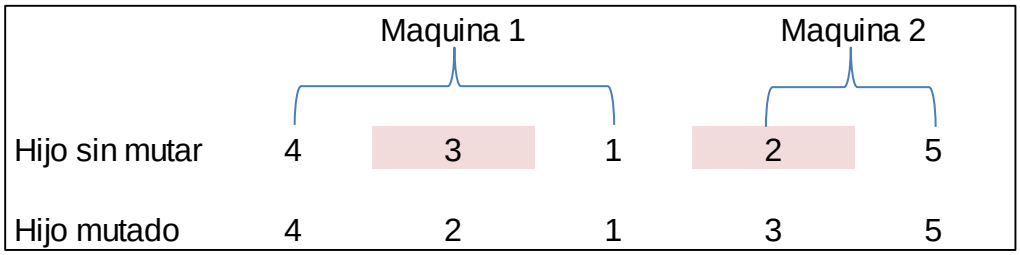


Figura 11: Representa el operador mutación, para ello se escogió al azar la posición número 2 en ambas máquinas, para así intercambiar los alelos en esa posición.

Fuente: Elaboración propia.

Para una mayor agilidad se implementó este algoritmo modificado en Matlab y así poder apreciar mejor su funcionalidad en este ambiente de trabajo, no obstante, para una mayor apreciación se hará una iteración en el programa diseñado en Matlab, y se adjuntará una imagen de la corrida, para notar el funcionamiento de los operadores genéticos.

Tabla 12

Ejemplo ilustrativo.

N° de trabajos	T. Procesado
1	45
2	30
3	60
4	100
5	90
6	64
7	20

Nota: En la tabla se pueden apreciar siete trabajos a programar en dos máquinas, con sus respectivos tiempos de procesamiento.

```
Command Window
ingrese el tamaño de la poblacion: 100
Elapsed time is 2.398248 seconds.
ingrese el numero de iteraciones: 1
ingrese la probabilidad de cruzamiento: 1
ingrese la probabilidad de mutacion: 1

individuo_1 =

     1     4     5     0     0     0     0     3     2     6     7     0     0     0   235


individuo_2 =

     3     5     4     6     7     2     0     1     0     0     0     0     0     0   364


individuo_1 =

     2     4     0     0     0     0     0     6     7     1     3     5     0     0   279


individuo_2 =

     0     0     0     0     0     0     0     4     3     7     6     2     1     5   409


Posible_padre =

     1     4     5     0     0     0     0     3     2     6     7     0     0     0   235
     2     4     0     0     0     0     0     6     7     1     3     5     0     0   279


Hijos_creados =

     1     7     2     3     5     0     0     6     4     0     0     0     0     0   245
     2     1     6     7     0     0     0     3     4     5     0     0     0     0   250


h_eleg_mut =

     1     7     2     3     5     0     0     6     4     0     0     0     0     0   245


hijo_mutado =

     1     4     2     3     5     0     0     6     7     0     0     0     0     0   325


Poblacion_temporal =

     1     7     2     3     5     0     0     6     4     0     0     0     0     0   245
```

Figura 12: Pantallazo de la salida de Matlab.

Fuente: Elaboración propia Mediante Software Matlab.

En la figura 12 señala cada secuencia los primeros siete números corresponden a los trabajos asignados a la primera máquina y los otros siete a la segunda máquina, y la última columna corresponde a la evaluación respectiva de la secuencia, teniendo en cuenta el *makespan*. Además, se le asignó una probabilidad del cien por ciento al operador de cruzamiento y mutación para observar su desarrollo.

Al aumentar el número de iteraciones, crecerá de tamaño la población conformada por los individuos de la segunda generación (hijos), entre mayor sea más facilidad se tendrá de escoger de todos ellos la combinación con el menor de tiempo de terminación, es decir; que su nivel de búsqueda aumentará para encontrar en el mayor de los casos óptimos locales.

Diseño del algoritmo genético en Matlab

Para el diseño del algoritmo genético, se basó en tres procesos fundamentales, propios de su solvencia teórica, es decir; que lo primero que se debe definir al momento de ejecutar este tipo de algoritmos de búsquedas: es una población, la cual va ser evaluada y pasada por los operadores genéticos, para su postrera elección del individuo que sobresalga ante toda la población, que sea el mejor bajo una función objetivo.

Por lo tanto, para mayor facilidad, se diseñó tres *Script*, considerando que el segundo debe ser alimentado por la información arrojada por el primero y el tercero por la información del segundo, aun así, se resalta que, para el primer, su sistema de alimentación será por medio de una tabla en Excel, para mejor destreza al momento de ingresar los datos de entrada primarios, que en este caso serán la cantidad de tareas a programar junto con su tiempo de terminación propios de cada tarea. Quedando de la siguiente manera conformado:

Tabla 13

Input y output del Script del Algoritmo genético.

Script	Descripción	Input	Output
Población_2maquina_ Paralelo.m	Crea la población a partir del número de tareas a programar en dos máquinas en paralelo idénticas, es decir; posibles soluciones.	• Cantidad individuos que conformaran la Población a crear. • Número de tareas con sus respectivos tiempos de procesamiento.	*Poblacion_generada :* Variable que almacena Varios individuos que conforman una población (posibles soluciones), más su tiempo de terminación máximo dé cada uno de ellos. (*Makespan*).
Operadores_geneticos. m	A partir de la población creada, se procede a la ejecución de los operadores genéticos utilizados: Selección, Cruce y Mutación.	• *Poblacion_generad a* • Número de iteraciones. • Probabilidad de Cruzamiento. • Probabilidad de Mutación.	*Poblacion_temporal :* Individuos de la siguiente generación (*hijos*).
Resultado_gentico.m	Escoge el mejor individuo de la *Población_temporal* , bajo una función objetivo, en este caso será la	• *Poblacion_tempora l*	• *Maquina_1*: las tareas asignada a la maquina Numero 1. • *Maquina_2*: las tareas asignada a

	minimización del tiempo de terminación máximo (*Makespan*), es decir; la secuencia que consuma menos tiempo en fabricarse.	la maquina Numero 2. • *Makespan*.

Para una mayor apreciación de la ejecución y funcionalidad del algoritmo genético, se debió precisar algunos parámetros y restricciones de manera general, que facilitaran al ajuste del algoritmo bajo un ambiente de máquinas en paralelos idénticas.

- Las maquinas se encuentran inicialmente disponibles.
- Las dos máquinas pueden trabajar en paralelo bajo un mismo ritmo.
- No se puede interrumpir el proceso, ni agregar una nueva tarea mientras este en curso, en caso tal lo amerite, se deberá correr nuevamente el algoritmo.
- No se tiene en cuenta los tiempos de *Set-Up*.
- No se pueden repetir las tareas en ambas máquinas.
- Las tareas asignadas en las maquinas se elaborarán de manera secuencial y sin interrupción.
- Cada individuo hace referencia a una posible solución, es decir: las posibles tareas asignadas en la maquina 1 y 2.
- Cada individuo será evaluado, por medio del tiempo que tardará en elaborarse cada una de las tareas, bajo el orden de asignación propio del individuo en las dos máquinas (*Makespan*).
- Para el cálculo del tiempo de terminación máximo – *Makespan*, se toma del tiempo de terminación mayor entre las dos máquinas, ya que se encontrarán trabajando en paralelo.

- Las unidades de evaluación del individuo y procesado de cada tarea, estarán expresados en unidades de tiempos.

Aun así, se considera los siguientes parámetros propios de los *Scripts* diseñados, para su buen funcionamiento, de los cuales se discriminan continuación:

Tabla 14

Parámetros para el Script: Algoritmo genético.

N°	*Script*	Parámetros Establecidos.
1	*Poblacion_2maquinas_paralelo*	• Se deberá ingresar los datos iniciales en el documento Datos.xlsx: en la primera columna estará ubicada la enumeración de las tareas organizadas de manera ascendente y en la segunda columna su respectivo tiempo de procesado. • El tamaño del vector a crear dependerá del número de tareas registradas en el documento de Excel. • Como ambas maquinas se encuentran disponibles inicialmente, se empezará asignando en la *maquina N° 1* de manera aleatoria. • Si en caso tal, el número aleatorio arrojado en cualquier instante para la *maquina N°1* es *Cero*, y quedan tareas por asignar, pasara a la *maquina N°2* para seguir el proceso de asignación de tareas, si no queda tareas por programar: se detiene el proceso y es considerado un individuo. • Al momento de crear el individuo, se procede a su

		evaluación, que será: el tiempo que consuma en elaborar todas las tareas bajo el orden del individuo. • Se termina el proceso: cuando se crea el número de individuos deseados por el programador. • Los operadores genéticos, se ejecutarán en el siguiente orden: *Selección, Cruce y Mutación*. Para luego ser almacenados en una población temporal (*Nueva generación*). • La *Selección* es de tipo *torneo*, es decir: que se escogerá de la población creada dos posibles padres al azar y se escogerá el mejor de ellos, según la función objetivo; el que menor tiempo consuma. • El número de iteraciones será definido por el programador, junto con los porcentajes probabilísticos de *Cruzamiento* y *Mutación*, se recomienda uno del 98% y 2% respectivamente. • El tipo de cruzamiento será de tipo doble corte, donde los hijos heredaran *información genética* de sus dos padres. • La tarea a mutar se escoge de manera aleatoria, y se compara si bajo su tiempo de terminación, si es afirmativo pasa el *hijo mutado* a la población temporal, de lo contrario pasa el *hijo sin mutar*. • En los procesos de *Cruzamiento* y *Mutación* se inspeccionan siempre al final del operador, si se encuentran tareas repetidas en el vector: será
2	*Operadores_genticos*	

		eliminada y colocada por la tarea faltante de manera aleatoria. • El proceso operativo es considerado elitista, ya que tiende a escoger y a darle participación al individuo más apto. • De todos los individuos que se encuentran en la población temporal, sacamos el que tengas el menor tiempo de terminación máximo, y así obtener la secuencia con menor tiempo de terminación.
3	*Resultado*	• Se visualiza las tareas asignadas en la *maquina número 1 y 2*, junto con el tiempo estimado para la terminación de ellas, los cuales estarán expresados en las mismas unidades de tiempo de las ingresadas en la tabla de Excel.

En los anexos se encuentra el código fuente diseñado en este proyecto, para la adaptación de un algoritmo genético bajo un ambiente de máquinas en paralelo idénticas.

Área de figurado estándar

Se hace indispensable el conocimiento del tiempo de los procesos internos del área de figuración estándar, sea cortar, doblar, traslados entre otros, ya que esto ayudaría a la aplicación de heurísticas para planificar, programar y controlar la producción en esta célula de trabajo. Ante este estudio se pudo observar los procesos que intervienen en el área de figurado estándar con sus respectivos tiempos. Cabe resaltar que el tiempo que tardan los operarios en la búsqueda de

puntas, (cortes) en el inventario es promedio y su desviación es considerablemente alta, debido al aumento de las puntas en inventario y a su desorganización.

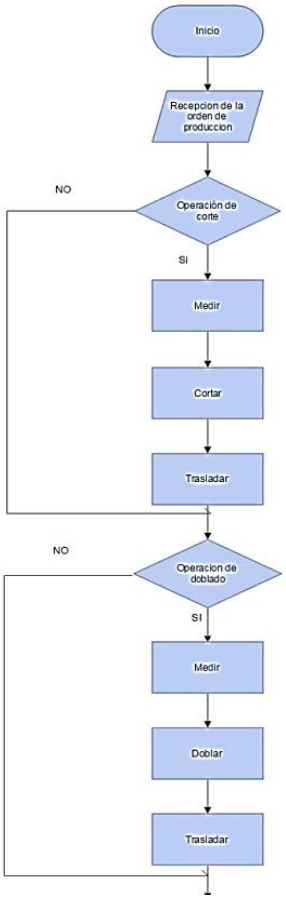


Figura 13: Diagrama de flujo del área de figurado estándar.

Fuente: Elaboración propia.

Tabla 15

Tiempos de los procesos en el área de cortado.

Operación	Tiempo (segundos)
Cortado	2
Medida	20
Traslado	20

Nota: Tiempo relacionado al área de cortado, las unidades del tiempo se encuentran en segundos.

Tabla 16

Tiempos de los procesos en el área de doblado.

Operación.	Tiempo (segundos)
Doblado	2
Medida	10
Traslado	5

Nota: Tiempo relacionado al área de doblado, las unidades del tiempo se encuentran en segundos.

Se mencionó que la actividad que restringe el área de figurado, conformando un cuello de botella, es el proceso de cortado, aunque no se comprueba bien en la tabla anterior, pero la demora en esta área se ve reflejada en los procesos subyacentes al de cortar, qué se requiere para realizar dicha tarea. Esto debido a la gran cantidad de cortés a realizar y el cálculo predeterminado del mismo, tomando con esto un tiempo considerable en el proceso.

Desarrollo del modelo.

Como se dijo anteriormente el área de cortado cuenta con un alto grado de responsabilidad en el proceso de esta área, por ende, se diseñó una heurística que tenga como finalidad mejorar y potencializar dicho proceso, ya que, si se logra, se estimaría una mejora en la línea de producción. Además, se aplicará la heurística de Johnson para la organización y planificación de

los grupos de figurado teniendo en cuenta el insumo a utilizar. En otras palabras, la heurística de Johnson establecería la secuencia de los grupos de figurados para disminuir los tiempos ocios y de entrega, y la otra heurística diría como hacer el figurado de manera particular. Para una mayor comprensión de la idea se realizará el siguiente ejemplo con las heurísticas diseñadas en Matlab.

Tabla 17

Ejemplo ilustrativo.

N°	Figura	Tipo de acero	Cantidad	Corte (mts)
1		# 4	10	12
2		# 4	10	10
3		# 4	5	9
4		# 4	10	4
5		# 4	5	3
6		# 4	10	8
7		# 4	16	2
1		# 5	10	9
2		# 5	14	8
3		# 5	20	4
4		# 5	5	3
1		# 7	10	11
2		# 7	12	10
3		# 7	9	8,8
4		# 7	1	4
5		# 7	5	4,1
6		# 7	7	3,2
7		# 7	8	3
8		# 7	10	2

Nota: Como se puede observar esta sección de la remisión viene organizada según el tipo de insumo, y en cada grupo de figurado se organiza de manera descendente según la longitud a cortar.

Como se pudo observar, los grupos de figurado se conforman según el insumo que necesitan. Las varillas estándar utilizadas para este ejemplo serán de 12 metros para cada insumo, aunque se encuentran de 9 y 6 metros. Como primer paso se desarrolla el algoritmo de unificación sin sobrantes, el cual unifica aquellos cortes del mismo tipo de insumo y que su suma de igual a la longitud de la varilla estándar; ya que se desea buscar aquellas posibles combinaciones de cortes que no generen sobrante alguno. Para ello se empezará tomando aquellos figurados que no necesitan ser cortados, es decir, la longitud del figurado que sea de 12 metros. Después se unifican los cortes en parejas, tríos, cuartetos, quintetos y sextos, y que su sumatoria de igual a la longitud de la varilla estándar. Se añade de ultimo la opción de unificar aquellos cortes que, gracias a su cantidad, su sumatoria sea igual a la longitud de la varilla estándar. Un ejemplo seria 6 cortes de 2 metros c/u, tomando la varilla de 12 metros, se cortaría en seis partes y se tendrían las unidades exigidas. Posterior al desarrollo de esté, si se presentan algunos cortes pendientes, aquellos que por más que se junten, ya sea la cantidad del mismo o con otro corte y no logren generar ningún sobrante con la varilla estándar, se procederá a usar el algoritmo de unificación con sobrantes. Su finalidad es generar unificaciones bajo el rango de permisibilidad de sobrantes, de modo que el programador deberá ingresar límites inferior y superior que indican lo máximo y mínimo que se aceptaría una punta refiriéndose a su longitud, de resto su funcionamiento es igual al anterior.

Como se puede apreciar, se debe ejecutar primero el algoritmo de unificación sin sobrantes y luego el algoritmo de unificación con sobrantes, para ello se ejecuta dicho lineamiento. En el ejemplo descrito anteriormente, se empezará con el insumo tipo 4, y veremos a continuación su resultado.

Tabla 18

Tareas a realizar del acero # 4.

Tarea	Tipo de acero	Cantidad	Corte (metros)
1	# 4	10	12
2	# 4	10	10
3	# 4	5	9
4	# 4	10	4
5	# 4	5	3
6	# 4	10	8
7	# 4	16	2

Nota: Se extrae el lote de producción con acero tipo # 4 para su desarrollo en el algoritmo diseñado en Matlab.

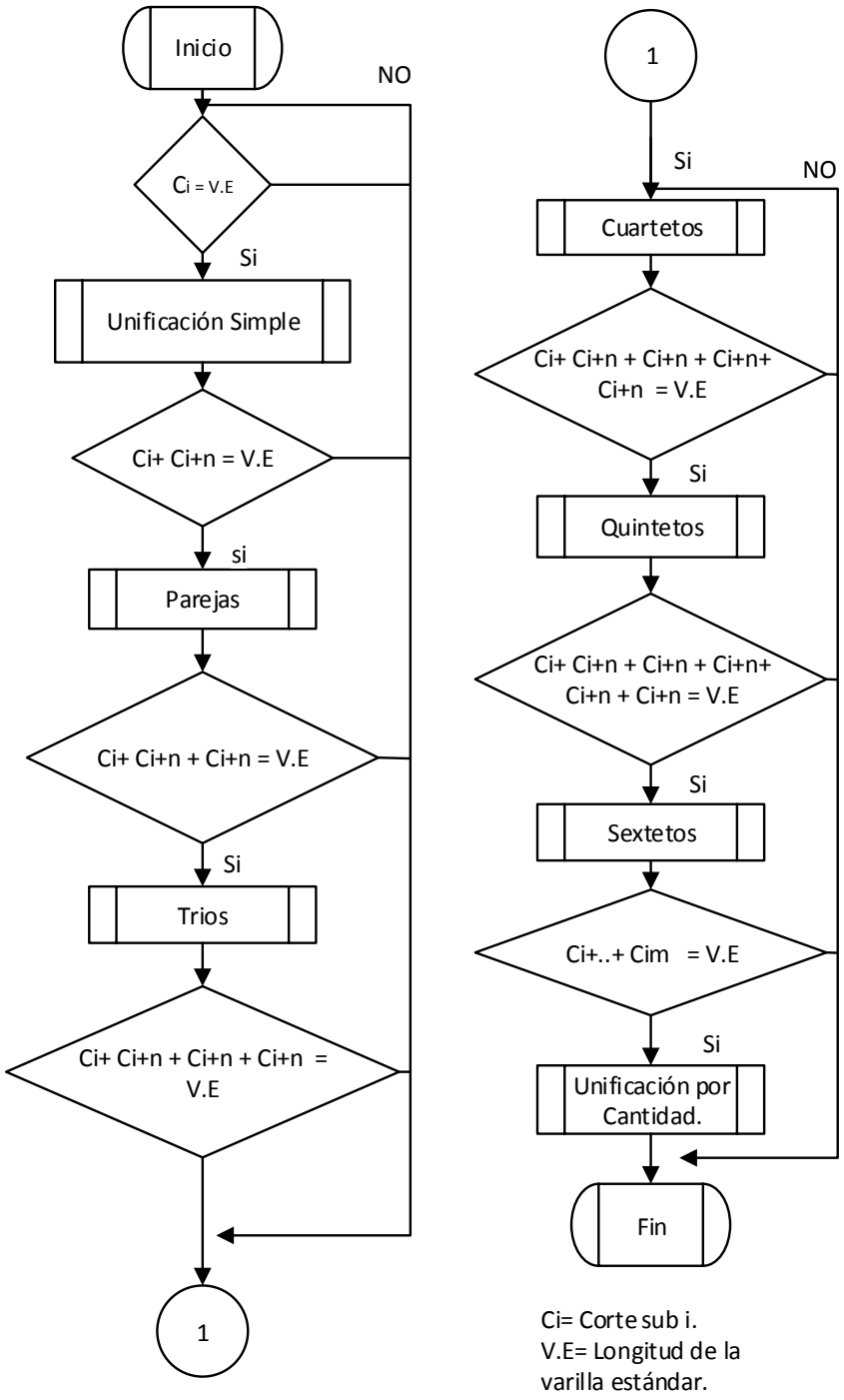


Figura 14: Diagrama de flujo de la heurística de cortes sin sobrantes.

Fuente: Elaboración propia.

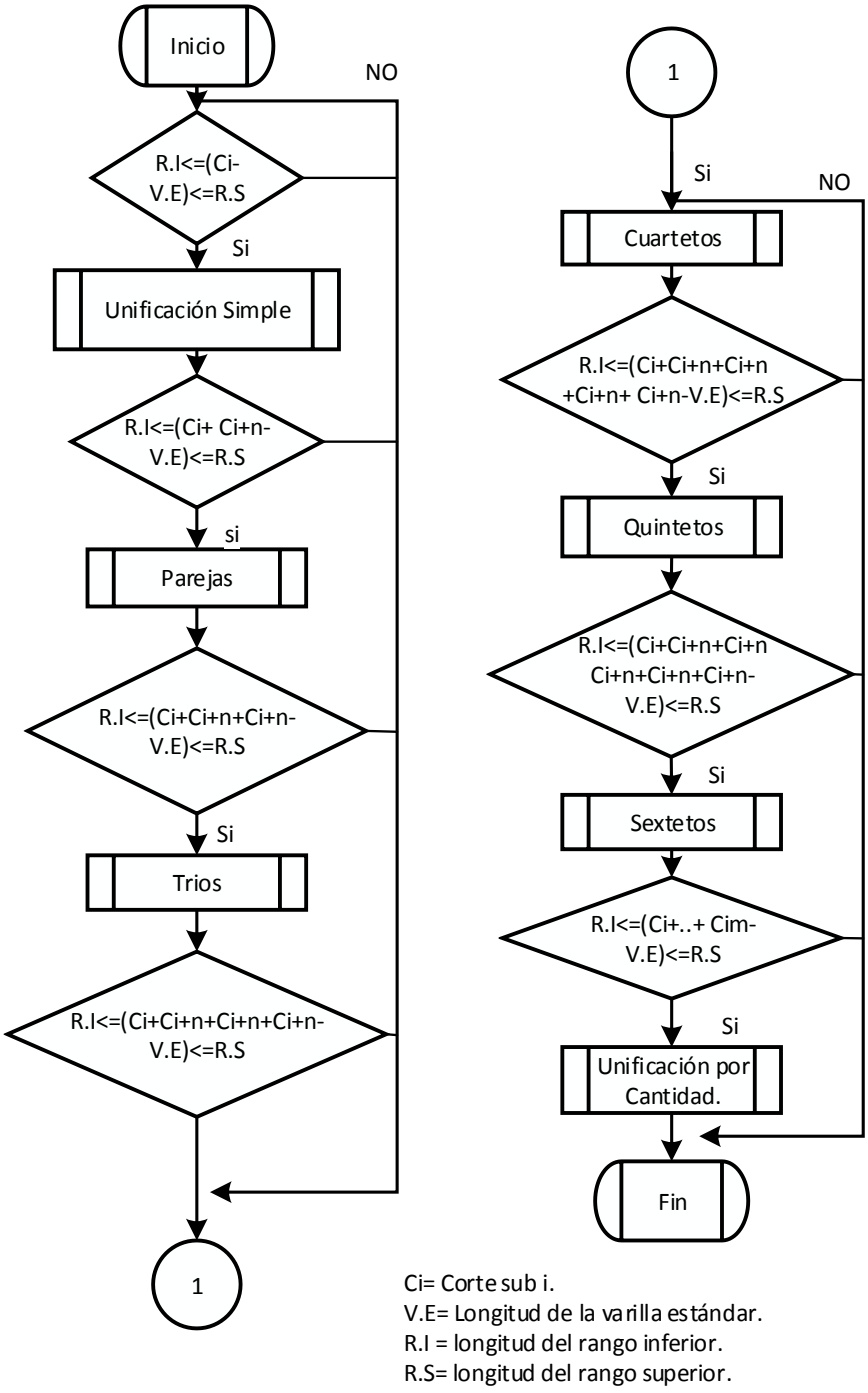


Figura 15: Diagrama de flujo de la heurística de cortes con sobrantes.

Fuente: Elaboración propia.

Resultado_1 | Resultado_2 | Resultado_Cantidad

1x1 struct with 5 fields

Field	Value	Min	Max
Trabajo_1	1	1	1
Tipo_Corte	12	12	12
Cantidad	10	10	10
Sobrante	0	0	0
Numero_varillas_utilizadas	10	10	10

Figura 16: Pantallazo de la salida de Matlab, indicando la variable resultado_1, son aquellos figurados que gracias a su longitud no necesitan ser cortados.

Fuente: Elaboración propia mediante Software Matlab.

Resultado_1 | Resultado_2 | Resultado_Cantidad

1x3 struct with 9 fields

Fields	Trabajo_1	Tipo_Corte	Cantidad	Sobrante	Trabajo_2	Tipo_Corte_2	Cantidad_2	Sobrante_2	Numero_varillas_utilizadas
1	2	10	10	0	7	2	16	6	10
2	3	9	5	0	5	3	5	0	5
3	4	4	10	0	6	8	10	0	10

Figura 17: Pantallazo de la salida de Matlab, indicando la variable resultado_2, son aquellos cortes agrupados en parejas y su suma es igual a la longitud de la varilla estándar.

Fuente: Elaboración propia mediante Software Matlab.

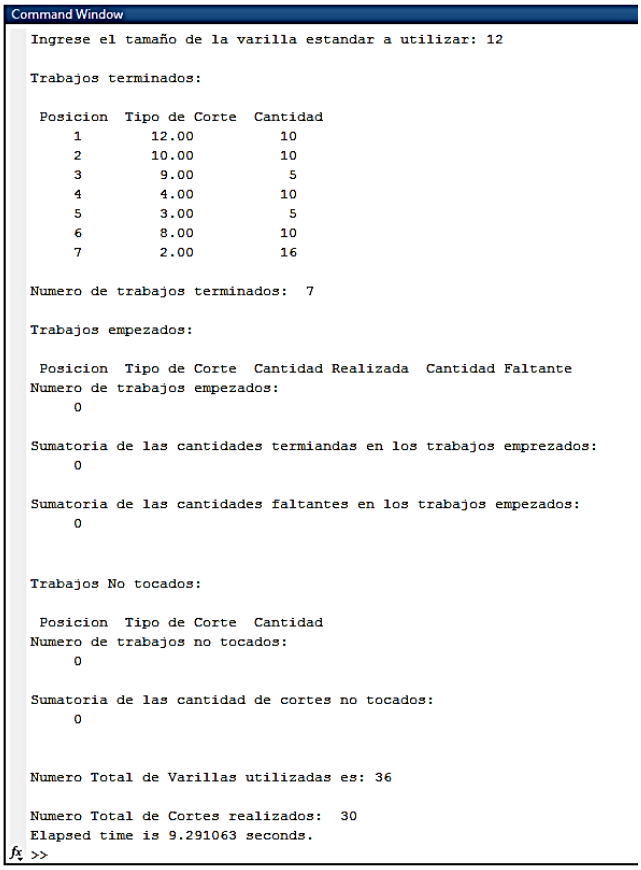
Resultado_1 | Resultado_2 | Resultado_Cantidad

1x1 struct with 6 fields

Field	Value	Min	Max
Trabajo	7	7	7
Tipo__Corte	2	2	2
Cantidad	6	6	6
Sobrante	0	0	0
Cantidad_cortes	5	5	5
Numero_varillas_utilizadas	1	1	1

Figura 18: Pantallazo de la salida de Matlab, indicando la variable resultado_cantidad, son aquellos cortes que, gracias a su cantidad, su totalidad es igual a 12 metros.

Fuente: Elaboración propia mediante Software Matlab.



```
Command Window
  Ingrese el tamaño de la varilla estandar a utilizar: 12

  Trabajos terminados:

   Posicion  Tipo de Corte  Cantidad
       1        12.00          10
       2        10.00          10
       3         9.00           5
       4         4.00          10
       5         3.00           5
       6         8.00          10
       7         2.00          16

  Numero de trabajos terminados:  7

  Trabajos empezados:

   Posicion  Tipo de Corte  Cantidad Realizada  Cantidad Faltante
  Numero de trabajos empezados:
       0

  Sumatoria de las cantidades termiandas en los trabajos emprezados:
       0

  Sumatoria de las cantidades faltantes en los trabajos empezados:
       0


  Trabajos No tocados:

   Posicion  Tipo de Corte  Cantidad
  Numero de trabajos no tocados:
       0

  Sumatoria de las cantidad de cortes no tocados:
       0


  Numero Total de Varillas utilizadas es: 36

  Numero Total de Cortes realizados:  30
  Elapsed time is 9.291063 seconds.
fx >>
```

Figura 19: Pantallazo del Command Windows de Matlab, mostrando con ellos los resultados generales de las unificaciones sin sobrantes hechas.

Fuente: Elaboración propia mediante Software Matlab.

Como se puede ver en la figura 18, gracias a las unificaciones no se generó para este caso sobrante, no obstante, se presentarán ocasiones cuando necesariamente se deben dejar cortes sin unificar, porque su suma no va a ser igual a la longitud del insumo utilizado. Por ahora, como ya se ha planificado la forma de cómo realizar los cortes con el insumo tipo # 4, expresados en las figuras 15, 16 y 17 dicha información será transmitida a los encargados del proceso de cortes para evitarles el cálculo de manera convencional, y así poder generarles la metodología

adecuada. Además de la información arrojada en la figura 18, se puede extraer el número de varillas estándar utilizadas en esta sección, logrando con ello el conocimiento exacto de la cantidad a utilizar en esta área para suministraselos de manera eficiente y exacta. También se puede deducir el número total de cortes realizados, ya que, gracias a esto, se obtiene la cantidad de mediciones y traslados realizados, alcanzando con ello el conocimiento del tiempo que tardarían realizando el proceso de corte en esta sección del figurado. Para ello se realizó una tabla que da a conocer las veces que se corta, mide y traslada cuando se realizan unificaciones de tipo único, pareja, trio, cuarteto, quinteto, sexteto y la opción cantidad.

Para obtener el tiempo que tardaría el proceso de doblado solo se tendría en cuenta el número de doblados que requiere la figura con sus respectivas mediciones y un solo traslado ya que de allí pasa al inventario de productos terminados.

Tabla 19

Número de cortes realizados en cada sección del programa.

Opciones de unificación.	Número de Cortes sin sobrantes.	Número de Cortes con sobrantes.
Único	Ninguno	1
Pareja	1	2
Trio	2	3
Cuarteto	3	4
Quinteto	4	5
Sexteto	5	6
Opción Cantidad	Depende de la cantidad estimada	Depende de la cantidad estimada

Nota: En la opción cantidad, depende de las cantidades agrupadas.

Tabla 20

Número de cortes realizados en cada sección del programa.

Tipos de figuras	Número de doblados	Número de mediciones	Número de Traslados
	2	1	1
	0	0	0
	1	1	1

Nota: Dependiendo de la figura, así será el número de dobles que se procederá a realizar.

Tabla 21

Tiempo empleado en el área de cortado para el lote de acero # 4.

Descripción	Resultado
Número total de cortes realizados	30
Tiempo total en cortar (segundos)	60
Número de traslados	30
Tiempo total en trasladar (segundos)	600
Número de mediciones	30
Tiempo total en medir (segundos)	600
Tiempo Total empleado en el proceso de corte (segundos)	1260
Tiempo Total empleado en el proceso de corte (minutos)	21

Nota: Resultados de las veces realizados por procesos en el área de cortado, para así obtener el tiempo que tardo cortando el lote de acero # 4, en este caso fueron 21 minutos

Tabla 22

Tiempo empleado en el área de doblado para el lote de acero # 4.

N°	Cantidad	N° de doblados por unidad	Tiempo Total de doblado (seg.)	N° de mediciones	Tiempo total de mediciones (seg.)	N° de Traslados	Tiempo total de traslados (seg.)
1	10	2	40	1	10	1	50
2	10	0	0	1	10	1	50
3	5	2	20	1	10	1	25
4	10	2	40	1	10	1	50
5	5	1	10	1	10	1	25
6	10	2	40	1	10	1	50
7	16	1	32	1	10	1	80
Total	66	10	182	7	70	7	330

Nota: Resultados de las veces realizados por procesos en el área de doblado, para así obtener el tiempo que tardo doblando el lote de acero # 4, en este caso fueron aproximadamente 10 minutos.

Tabla 23

Tiempo total empleado.

Proceso	Tiempo de procesamiento para el acero # 4 (min.)

Cortado	21
Doblado	10
TOTAL	**31**

Nota: Tiempo total empleado para el lote de producción del acero #4 es de 31 minutos.

El tiempo de producción estimado para el acero tipo # 4 con su respectiva figuración es de 31 minutos. Es preciso tener en cuenta que este es el tiempo neto de producción, sin meter los atrasos, demoras y descansos que se presenten en la operación. De manera similar se procede con el acero # 5, el resultado arrojado por el programa fue el siguiente.

Tabla 24

Tareas a realizar del acero # 5.

Tipo de acero	Cantidad	Corte (metros)
# 5	10	9
# 5	14	8
# 5	20	4
# 5	5	3

Nota: Se extrae el lote de producción con acero tipo # 5 para su desarrollo en el algoritmo diseñado en Matlab.

Resultado_2 | Resultado_Cantidad

1x2 struct with 9 fields

Fields	Trabajo_1	Tipo_Corte	Cantidad	Sobrante	Trabajo_2	Tipo_Corte_2	Cantidad_2	Sobrante_2	Numero_varillas_utilizadas
1	1	9	10	5	4	3	5	0	5
2	2	8	14	0	3	4	20	6	14

Figura 20: Pantallazo de la salida de Matlab, indicando la variable resultado_2, son aquellos cortes agrupados en parejas y su suma es igual a la longitud de la varilla estándar.

Fuente: Elaboración propia mediante Software Matlab.

Resultado_2 | Resultado_Cantidad

1x1 struct with 6 fields

Field	Value	Min	Max
Trabajo	3	3	3
Tipo__Corte	4	4	4
Cantidad	6	6	6
Sobrante	0	0	0
Cantidad_cortes	4	4	4
Numero_varillas_utilizadas	2	2	2

Figura 21: Pantallazo de la salida de Matlab, indicando la variable resultado_cantidad, son aquellos cortes que, gracias a su cantidad, su totalidad es igual a 12 metros.

Fuente: Elaboración propia mediante Software Matlab.

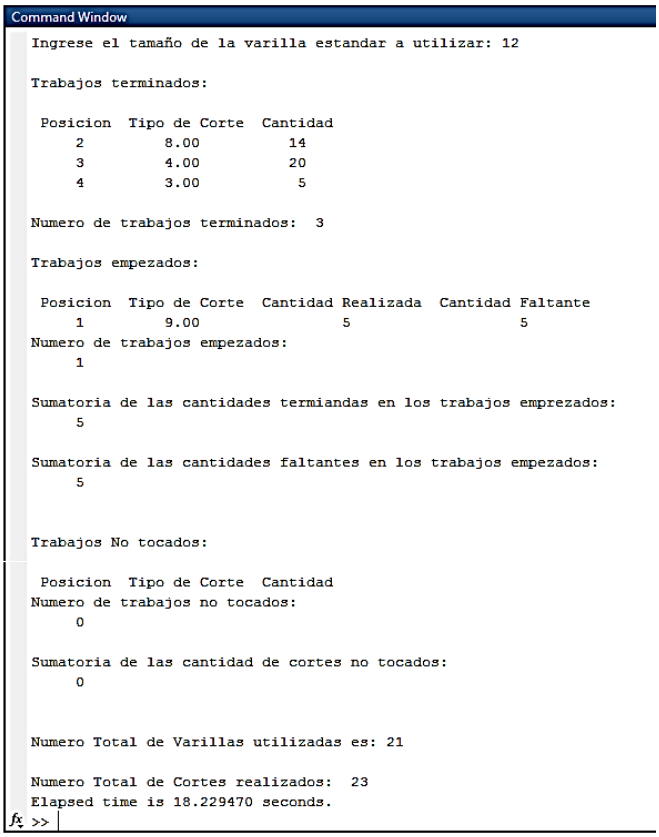


Figura 22: Pantallazo del Command Windows de Matlab, mostrando con ellos los resultados generales de las unificaciones sin sobrantes hechas.

Fuente: Elaboración propia mediante Software Matlab.

Como se aprecia en la figura 21, notamos que quedó un trabajo empezado, o sea, que en las unificaciones sin sobrantes se realizó solo parte de él, faltando 5 cortes de 9 metros.

Previo a la ejecución del algoritmo de unificación con sobrantes, es recomendable que antes de ejecutarlo se evalúen otras opciones, en compañía de los encargados de la elaboración de las remisiones. Una de ella sería la utilización para este caso de varillas estándar de 9 metros, la

reducción de alguna punta existente en inventario o la disminución o aumento de algunas decima o centímetros del figurado o longitud del corte. Pero para este ejemplo, ver el funcionamiento y los resultados arrojados por el algoritmo de unificación con sobrantes, y se estimará como rango de permisibilidad, puntas de 3 metros.

Resultado_1

1x1 struct with 6 fields

Field	Value	Min	Max
Trabajo_1	1	1	1
Tipo_Corte	9	9	9
Cantidad	5	5	5
Numero_varillas_utilizadas	5	5	5
Longitud_punta	3	3	3
Cantidad_puntas_sobrantes	5	5	5

Figura 23: Pantallazo de la salida de Matlab La variable Resultado_1 hace alusión aquellos cortes realizados una vez, teniendo con esto el coste deseado más una punta (sobrante).

Fuente: Elaboración propia mediante Software Matlab.

```
Command Window
  Ingrese el tamaño de la varilla estandar a utilizar: 12
  ingrese el Rango de Permisibilidad superior respecto a la longitud del sobrante: 3
  ingrese el Rango de Permisibilidad Inferior respecto a la longitud del sobrante: 1

  Trabajos terminados:

   Posicion  Tipo de Corte  Cantidad
       1         9.00          10
       2         8.00          14
       3         4.00          20
       4         3.00           5

  Numero de trabajos terminados:  4

  Trabajos empezados:

   Posicion  Tipo de Corte  Cantidad Realizada  Cantidad Faltante
  Numero de trabajos empezados:
       0

  Sumatoria de las cantidades termiandas en los trabajos emprezados:
       0

  Sumatoria de las cantidades faltantes en los trabajos empezados:
       0


  Numero Total de Varillas utilizadas es: 5

  Numero Total de de longitudes sobrantes es: 5

  Numero Total de Cortes realizados:  5
fx >>
```

Figura 24: Pantallazo del Command Windows de Matlab, mostrando con ellos los resultados generales de las unificaciones sin sobrantes hechas.

Fuente: Elaboración propia mediante Software Matlab.

Como se esperaba el resultado arrojó el número de cortes realizados en esta sección y la cantidad de puntas sobrantes con sus respectivas medidas; en este caso fueron cincos puntas sobrantes cada una de 3 metros. Sabiendo el número cortes totales que se requirió para el proceso de cortado con el acero # 5, se procederá al cálculo del tiempo de producción neto.

Tabla 25

Tiempo empleado en el área de cortado para el lote de acero # 5.

Descripción	Resultado
Número total de cortes realizados	28
Tiempo total en cortar (seg.)	56
Número de traslados	28
Tiempo total en trasladar (seg.)	560
Número de mediciones	28
Tiempo total en medir (seg.)	560
Tiempo Total empleado en el proceso de corte (seg.)	1176
Tiempo Total empleado en el proceso de corte (min.)	19,6

Nota: Resultados de las veces realizados por procesos en el área de cortado, para así obtener el tiempo que tardo cortando el lote de acero # 5, en este caso fueron aproximadamente 20 minutos.

Tabla 26

Tiempo empleado en el área de doblado para el lote de acero # 5.

N°	Cantidad	N° de doblados por unidad	Tiempo Total de doblado (seg.)	N° de mediciones	Tiempo total de mediciones (seg.)	N° de Traslados	Tiempo total de traslados (seg.)
1	10	1	20	1	10	1	50
2	14	2	56	1	10	1	70
3	20	2	80	1	10	1	100
4	5	1	10	1	10	1	25
Total	49	6	166	4	40	4	245

Nota: Resultados de las veces realizados por procesos en el área de doblado, para así obtener el tiempo que tardo doblando el lote de acero # 5, en este caso fueron aproximadamente 8 minutos.

Tabla 27

Tiempo total empleado.

Proceso	Tiempo de procesamiento para el acero # 5 (min.)
Cortado	20
Doblado	8
TOTAL	**28**

Nota: Tiempo total empleado para el lote de producción del acero # 5 es de 28 minutos.

Sólo nos haría falta el figurado del acero tipo número 7 qué se realizará a continuación, obteniendo con esto la metodología para realizar el proceso de cortado incluyendo su tiempo neto de producción.

Tabla 28.

Tareas a realizar del acero # 7.

N°	Tipo de acero	Cantidad	Corte (metros)
1	# 7	10	11
2	# 7	12	10
3	# 7	9	8,8
4	# 7	1	4
5	# 7	5	4,1
6	# 7	7	3,2
7	# 7	8	3
8	# 7	10	2

Nota: Se extrae el lote de producción con acero tipo # 5 para su desarrollo en el algoritmo diseñado en Matlab.

Tabla 29.

Tiempo empleado en el área de cortado para el lote de acero # 7.

Descripción	Resultado
Número total de cortes realizados	43
Tiempo total en cortar (seg.)	86
Número de traslados	43
Tiempo total en trasladar (seg.)	860

Número de mediciones	43
Tiempo total en medir (seg.)	860
Tiempo Total empleado en el proceso de corte (seg.)	1806
Tiempo Total empleado en el proceso de corte (min.)	30,1

Tabla 30

Tiempo empleado en el área de doblado para el lote de acero # 7.

N°	Cantidad	N° de doblados por unidad	Tiempo Total de doblado (seg.)	N° de mediciones	Tiempo total de mediciones (seg.)	N° de Traslados	Tiempo total de traslados (seg.)
1	10	0	0	1	10	1	5
2	12	2	48	1	10	1	5
3	9	2	36	1	10	1	5
4	1	2	4	1	10	1	5
5	5	1	10	1	10	1	5
6	7	0	0	1	10	1	5
7	8	1	16	1	10	1	5
8	10	2	40	1	10	1	5
Total	62	10	154	8	80	8	40

Tabla 31.

Tiempo total empleado.

Proceso	Tiempo de procesamiento para el acero # 7 (min.)
Cortado	30
Doblado	5
TOTAL	**35**

Nota: Tiempo total empleado para el lote de producción del acero # 7 es de 28 minutos.

De lo anterior se puede concretar la siguiente tabla.

Tabla 32.

Tiempo total empleado en cada lote de producción.

Sección según el tipo de insumo.	Tiempo que tarda en el proceso de cortado (min).	Tiempo que tarda en el proceso de doblado(min).	**Total.**
Figurado del acero # 4	21	10	**31**
Figurado del acero # 5	20	8	**28**
Figurado del acero # 7	30	5	**35**
Total	**71**	**23**	

Ya conociendo los tiempos de producción netos, se puede derivar una secuencia para la asignación de los grupos de figurado, en este caso serían los tres grupos apreciados en la tabla 31. Es preciso distinguir que al modificar el orden o secuencia de asignación es esta área de trabajo, puede variar en algunas ocasiones el tiempo de producción de todo el figurado estándar a procesar y el tiempo ocio. Para ello se asignaron los trabajos, calculando el tiempo de terminación máximo y el tiempo ocio de la línea de producción.

Tabla 33.

Secuencia tomada al azar.

Secuencia	Sección según el tipo de insumo
3	Figurado del acero # 7
1	Figurado del acero # 4
2	Figurado del acero # 5

Tabla 34.

Evaluación de la secuencia tomada al azar.

Secuencia	Tiempo ocio (min.)	Tiempo de terminación máximo (*makespan*) en minutos
3-1-2	64	79

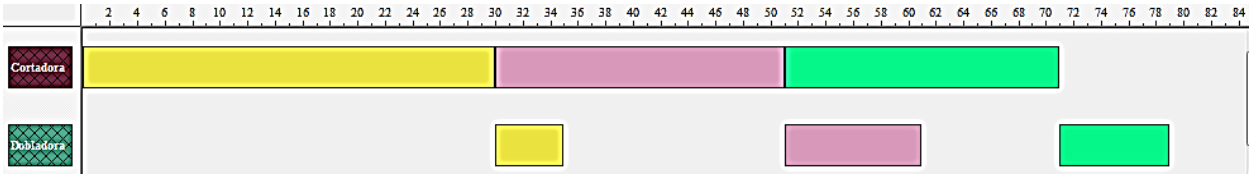


Figura 25: Grafica de Gantt realizado en Lekin®, mostrando los tiempos de ejecución y ocios de la secuencia tomada.

Fuente: Elaboración propia Mediante Software Lenkin.

Al importar el orden se debe poner en ejecución una heurística capaz de dar buenas soluciones en estos ambientes de trabajo, pudiéndose afirmar que el tiempo ocio tendería a disminuir cuando el tiempo de terminación decrece; pero aun así lograr un "optimo" en este caso se hace imposible, ya que se tendría que reducir a cero los tiempos ocios de las estaciones de trabajos. Como se vio en el primer trabajo asignado, la segunda maquina o puesto de trabajo debe esperar la culminación de la labor en la primera máquina. Así la primera estación de trabajo permanece improductiva, mientras se procesa el último trabajo en la segunda máquina. (Sippper & Bulfin, 1998, pág. 444). A esto se le suman otros tiempos ocios intermedios, no obstante, la reducción de estos, permitiría una línea de producción más eficiente. Para ello existen varios métodos clásicos, que debido a su idoneidad, se empleará la heurística de Johnson (JOHNSON, 1953), la cual ayudará a la disminución en los tiempo muertos al inicio (maquina 2) y al final (maquina 1), que consistió en poner los trabajos que poseían menor tiempo de ejecución en la maquina 1, colocarlos al inicio y en la maquina 2 al final, para así poder lograr una disminución en los tiempos ocios. Al lograr esto se tendría a su vez una mengua en el tiempo de terminación

máximo (*makespan*). Para ello se diseñó la heurística en Matlab, y la secuencia obtenida fue la siguiente:

Tabla 35.

Evaluación de la secuencia arroja con la heurística de Johnson.

Secuencia	Tiempo ocio del proceso de doblado (min.)	Tiempo de terminación máximo (*makespan*) en minutos
1-2-3	58	76

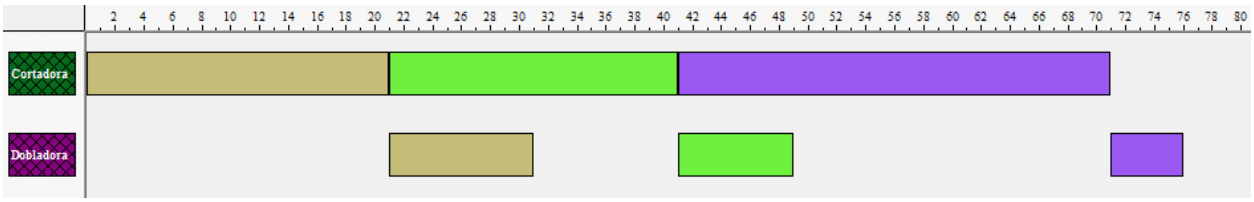


Figura 26: Grafica de Gantt realizado en Lekin®, mostrando los tiempos de ejecución y ocios de la secuencia arrojada con la heurística de Johnson

Fuente: Elaboración propia Mediante Software Lenkin.

.

El tiempo ocio disminuyó en 6 minutos, y el tiempo de terminación en 3 minutos; debido a la poca desviación en los tiempos de fabricación en el proceso de cortado y doblado, y a la cantidad de grupos de figuraciones. La diferencia no fue mucha, pero entre más varíen los tiempos de ejecución de los lotes de producción y aumente el número de trabajos se podría apreciar mejor la factibilidad de la metodología, algo notable en las ordenes de producción emitidas a esta área, ya que puede variar de 1 hasta 6 lotes de producción.

Diseño de las heurísticas para el área de figurado estándar en Matlab.

En este caso, se diseñaron tres *Script*, de los cuales se conectan y sirven de información de entrada para la ejecución del algoritmo, como se puede observar en la siguiente tabla:

Tabla 36.

Descripción de los Scripts para el área de figuración estándar.

Script.	**Descripción.**
Uniones_Cortes.m	La finalidad de este algoritmo es poder unir todos aquellos cortes que su suma sea igual a la longitud de la varilla estándar, para lograr como primera instancia no generar puntas o sobrantes.
Accion_Cortar.m	La finalidad de este algoritmo es poder unir todos aquellos cortes que su suma menos la longitud de la varilla estándar (insumo), sea igual al rango de permisibilidad establecido, es decir; que no se pase del rango de aceptación de longitudes de sobrantes.
Heurística_johnson.m	Después de Saber el tiempo que se toma en cortar y doblar cada tipo de figurado, se procede a programar los trabajos por medio de la heurística de Johnson.

Tabla 37

Input de los Scripts para el área de figuración estándar.

Script	**Input**
Uniones_Cortes.m	• Longitud de la varilla estándar a utilizar para los cortes. • Tipos de cortes con sus respectivas cantidades a cortar.
Accion_Cortar.m	• Resultado (1).xls: documento en formato Excel arrojado después de la ejecución del algoritmo anterior, mostrando las referencias, tipos de cortes y cantidades, que debido a su longitud no se pudieron unir sin generar sobrantes. • Longitud de la varilla estándar.

Heurística_johnson.m	• Rango de Permisibilidad superior respecto a la longitud del sobrante. • Rango de Permisibilidad Inferior respecto a la longitud del sobrante • Tipos de figurados, con su respectivo tiempo del proceso de cortado y doblado.

Tabla 38.

Output de los Scripts para el área de figuración estándar.

Script	**Output**
Uniones_Cortes.m	• Resultado (1).xls: Documento en formato Excel arrojado después de la ejecución del algoritmo, mostrando las referencias, tipos de cortes y cantidades, que debido a su longitud no se pudieron unir sin generar sobrantes. • Resultado_1: Numero de cortes que, debido a su longitud y la varilla estándar utilizada, no necesitan ser cortados ya que tienen la misma longitud de la varilla estándar. • Resultado_2: Número de parejas unificadas que su suma sea igual a la longitud de la varilla estándar. • Resultado_3: Número de tríos unificados, que su suma sea igual a la longitud de la varilla estándar. • Resultado_4: Número de cuartetos unificados, que su suma sea igual a la longitud de la varilla estándar. • Resultado_5: Número de Quintetos unificados, que su suma sea igual a la longitud de la varilla estándar. • Resultado_6: Número de sextetos unificados, que su suma sea igual a la longitud de la varilla estándar. • Resultado_cantidad: Numero de unificaciones formadas a partir de la cantidad del corte; que su suma de igual a la longitud de la

Accion_Cortar.m	varilla estándar. • Número de cortes: Satisfechos, Empezados, No tocados. • Cantidad de insumos utilizados. • Cantidad de Cortes realizados y faltantes. • Resultado_1: Numero de cortes que, debido a su longitud y la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_2: Numero de pares unificados que, debido a la suma de su longitud y la diferencia de ello con la longitud de la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_3: Numero de tríos unificados que, debido a la suma de su longitud y la diferencia de ello con la longitud de la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_4: Numero de cuartetos unificados que, debido a la suma de su longitud y la diferencia de ello con la longitud de la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_5: Numero de quintetos unificados que, debido a la suma de su longitud y la diferencia de ello con la longitud de la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_6: Numero de sextetos unificados que, debido a la suma de su longitud y la diferencia de ello con la longitud de la varilla estándar utilizada, al momento de ser cortados, la longitud de la punta o sobrantes se encuentra en el rango de permisibilidad. • Resultado_Cantidad: Indica el número de cortes que, gracias a su

	cantidad, la suma de ello menos la longitud de la varilla estándar se encuentran bajo el rango de permisibilidad de sobrantes. • Número de cortes: Satisfechos, Empezados y No tocados. • Cantidad de insumos totales utilizados. • Cantidad de Cortes totales realizados y Faltantes.
Heurística_johnson.m	• La secuencia estimada con su tiempo de terminación máximo.

En los anexos se encuentra el código fuente diseñado en este proyecto, los algoritmos que hacen referencia a las heurísticas diseñadas para la planificación y programación de tareas al área de figuración estándar.

Modelo para la organización y planificación de las remisiones en cola.

Gracias a las dos propuestas anteriores se puede lograr la obtención del tiempo de producción de la remisión, ya que esta se encuentra dividida en dos partes; la primera compete al área de figurado asistida por computador y la otra al área de figuración estándar. Cada una de ellas por medio de la aplicación de algoritmo genético en máquinas en paralelo y las heurísticas mencionadas anteriormente, ayuda al conocimiento del tiempo de ejecución de la remisión, y cada una de ellas vienen con su tiempo de entrega, teniendo con esto dos variables fundamentales, que facilitarían el cálculo para escoger la mejor manera de organizar las remisiones en cola. La metodología propuesta para la organización de remisiones sería la aplicación de reglas de despacho, se escogieron las más utilizadas, y se aplicará cada una de ellas, consiguiendo con esto evaluar los siguientes criterios:

- Número de remisiones que no se entregarán a tiempo.
- Número de remisiones adelantadas.
- Tardanza máxima y promedio de las remisiones programadas.
- Adelanto máxima y promedio de las remisiones programadas.
- Tiempo de terminación de todas las remisiones.
- Tiempo flujo de cada remisión.

Como se puede observar, las reglas de despacho, llamada en algunas ocasiones reglas de prioridad, utilizadas en este caso serían las expresadas en la tabla 35 (CHASE, JACOBS, & AQUINALO, 2009, pág. 627), para una mayor apreciación de su utilidad se expondrá el siguiente ejemplo.

Tabla 39

Reglas de despacho utilizadas en este modelo.

Regla de prioridad	Nomenclatura	Descripción
primero en entrar, primero en trabajarse, *(first-come, first-served)*	FCFS	Los pedidos se ejecutan en el orden en que llegan al departamento.
Tiempo de operación más breve, (*shortest operating time*).	SOT	Ejecutar primero el trabajo con el tiempo de terminación más breve, luego el siguiente más breve, etc.
Primero el plazo más próximo, *(earliest due date first).*	EDD	Se ejecuta primero el trabajo que antes se venza.
Tiempo ocioso restante, *(slack time remaining).*	STR	Se calcula como el tiempo que queda antes de que se venza el plazo menos el tiempo restante de procesamiento. Los pedidos con menor tiempo ocioso restante se ejecutan primero.

Tabla 40

Ejemplo ilustrativo.

Remisiones #	Fecha de entrega	Tiempo de procesado (horas)
1	3 días	7
2	2 días	5
3	2 días	4
4	1 día	3
5	1 día	5

Es necesario que los tiempos de entrega y de procesamiento manejen las mismas unidades de tiempo, para ello se pasarán los días de las fechas de entregas en horas, y con ello se tendrá en cuenta que un día equivale a un jornal laboral (8 horas diarias).

Tabla 41

Tabla de conversión de las unidades de tiempo.

Fecha de entrega en días	Fecha de entregas en horas
3	24
2	16
2	16
1	8
1	8

Posterior a ellos se ejecutará el programa diseñado en este proyecto, para que evalúe con cada regla de despacho los criterios mencionados anteriormente.

Tabla 42

Resultados.

R.D.	Cmax.	N° de tareas tardíos	N° de tareas adelantados	Tardanza máx.	Adelanto máx.	Tardanza prom.	Adelanto prom.	Tiempo flujo.
FCFS	24	2	3	16	17	5,4	4,2	78
SOT	24	1	4	9	9	1,8	3,6	63
EDD	24	1	4	1	5	0,2	1,6	65
STR	24	1	4	1	3	0,2	1,2	67

Nota: Los resultados arrojados por el algoritmo realizado en Matlab.

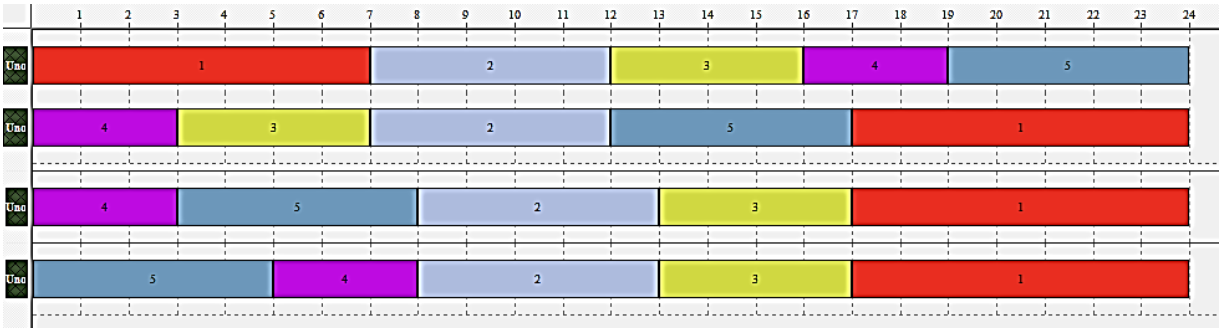


Figura 27: Gráfico de Gantt realizado en Lekin®.

Fuente: Elaboración propia Mediante Software Lenkin.

La tabla 38 muestra el resultado del programa propuesto en este proyecto, en ella se nota que las reglas SOT, EDD y STR tiene proyectado el atraso de una remisión, mientas que en la primera nos muestra dos atrasos, por ende, la secuencia a escoger se encontrará en estas tres opciones. Pero examinando con mayor detenimiento la tabla, se ve que con la secuencia STR se estimará la menor tardanza y adelanto máximo, con respecto las demás, indicando que sería la secuencia con menor índice de incumplimiento y productos terminados en inventario. Si se logra establecer una correcta sincronización con el proceso de despacho de remisiones, se tendría

solamente que, por mucho, las cuatro remisiones permanecerían en inventario 3 horas, tiempo acertado en este ejemplo para el cargue y despacho del producto. Con respecto a la remisión número 3 se terminará de procesar en 1 hora después de lo pactado, trabajando 8 horas diarias, de modo que para su cumplimiento solo se establecería 1 hora extra.

En este caso la mejor secuencia fue STR, pero ello no indica que siempre será así, ya que, al variar el número de remisiones, fechas de entrega y tiempo de producción varían los resultados por lo tanto se hace pertinente la evaluación previa ante estas cuatro reglas de despacho para así escoger la mejor.

Diseño de las reglas de despachos en Matlab.

Cuando se tiene varias remisiones en cola, y el tiempo de procedimiento de cada una de ellas, gracias a la ejecución de los algoritmos anteriormente mencionados, se procede a ejecutar el algoritmo que permita la planificación de las remisiones en cola. Para ello se debe conocer el tiempo de entrega y procesado de cada remisión en cola. En este caso se diseñaron las siguientes reglas de despacho en Matlab:

Tabla 43.

Reglas de despacho diseñadas en Matlab.

Regla de despacho.	Descripción.
FCFS "Firts-Come, First-Served"	Organizan las remisiones según su orden de llegada.
SOT "shortest operating time"	Organiza las remisiones de menor a mayor según su tiempo de producción.
EDD "earliest due date first"	Organizan las remisiones según su fecha de vencimiento.
STR "slack time remaining"	Organizan las remisiones según su tiempo ocio restante.

Se diseñó por cada regla de despacho un *Script.* Y uno adicional que tiene por nombre: *Reglas_Despacho.m*, que se encarga de evaluar cada regla de despacho las siguientes variables:

- Tiempo de terminación de todas las tareas.
- Número de trabajos tardíos.
- Número de trabajos adelantados.
- Tardanza máxima
- Adelanto máximo
- Tardanza promedio.
- Adelanto promedio.
- Tiempo flujo.

Como información de entrada, se requerirá precisar en un documento de Excel, las remisiones en cola, con su respectivo tiempo de procesamiento y fecha de entrega, las mismas unidades de tiempo de procedimiento de la remisión debe ser igual a las unidades de tiempo de las fechas de entregas, para así poder evaluar y apreciar su funcionalidad.

En los anexos se encuentra el código fuente diseñado en este proyecto, para el diseño y aplicación de las reglas despacho para la programación de las remisiones en cola.

Resultados y análisis

Área de figuración asistida por computador.

Para una mayor apreciación del nivel de búsqueda, optimización, y de verificar su confiabilidad, se debió comparar los resultados arrojados por el algoritmo genético con un programa (Lekin®) desarrollado por los docentes: Michael L. Pinedo, Xiuli Chao y Joseph Leung, de Stern School of Business de la Universidad de Nueva York. Lekin® es un software libre que permite la programación y planificación de la producción por medio de reglas de despacho y algoritmo de búsqueda en diferentes ambientes de trabajo. Para una muestra se le programó los sietes trabajos de la tabla 12 en dos máquinas en paralelo idénticas con sus respectivos tiempos de procesamientos. Primero se aplicaron algunas reglas de prioridad y el resultado arrojado por Lekin® fue el siguiente.

Tabla 44

Aplicación de las reglas de prioridad

Regla de prioridad	Característica de la regla	Secuencia obtenida para la maquina 1	Secuencia obtenida para la maquina 2	Tiempo de terminación máximo
EDD	Desarrollar primero el pazo más próximo.	Trabajos: 1, 4 y 6	Trabajos: 2, 3, 5 y 7	209
FCFS	Primero en llegar, primero en desarrollarse.	Trabajos: 1, 4 y 6	Trabajos: 2, 3, 5 y 7	209
CR	Porción critica.	Trabajos: 1, 3 y 5	Trabajos: 2, 7, 6 y 4	214
LPT	Organiza los trabajos de mayor a menor según el tiempo de procesamiento de cada tarea.	Trabajos: 4, 3, 2 y 7	Trabajos: 5, 6 y 1	210

SPT	Organiza los trabajos de menor a mayor según el tiempo de procesamiento de cada tarea.	Trabajos: 7, 1, 6 y 4	Trabajos: 2, 3 y 5	229

Nota: Resultados obtenidos con la ayuda del software Lekin®

Para la primera regla de despacho, el software estima que en el orden asignado se encuentran organizados, de manera ascendente los tiempos de entrega. Del resultado arrojado se puede apreciar que, al ser evaluado según el *makespan*, la mejor opción fueron las dos primeras reglas tardando 209 unidades de tiempo. Para una evaluación más exhaustiva el programa permite la búsqueda de una secuencia por medio de un algoritmo que establece como función objetivo el *makespan*, la secuencia obtenida fue la siguiente.

Tabla 45

Aplicación del algoritmo de búsqueda.

Secuencia para maquina 1	Secuencia para maquina 2	Makespan
Trabajos: 2, 6, 5, 7	Trabajos: 4, 1, 3	205

Nota: Resultados obtenidos con la ayuda del software Lekin®

Hasta este momento Lekin® arrojó su mejor resultado por medio de su algoritmo de búsqueda. A continuación, se muestra el resultado conseguido con el algoritmo genético diseñado en este proyecto.

```
Command Window
  ingrese el tamaño de la poblacion: 100000
  Elapsed time is 11.859678 seconds.
  ingrese el numero de iteraciones: 1000
  ingrese la probabilidad de cruzamiento: 0.98
  ingrese la probabilidad de mutacion: 0.1

  Maquina_1 =

       4     1     3     0     0     0     0


  Maquina_2 =

       7     2     5     6     0     0     0


  Tiempo_terminacion =

     205

fx >>
```

Figura 28: Pantallazo de la salida de Matlab que hace referencia al desarrollo del algoritmo genético diseñado en este propuesto.

Fuente: Elaboración propia Mediante Software Matlab.

Como se puede observar el *Makespan* del algoritmo genético diseñado fue menor que las de las reglas de prioridad e igualó al algoritmo de búsqueda arrojado por Lekin®. Gracias a esto se puede validar y comprobar la factibilidad del programa diseñado en este proyecto, ya que el tiempo de ejecución fue considerable.

Área de figuración estándar.

El objetivo es poder establecer secuencias de pasos confiables al momento programar y controlar la elaboración en el área de figuración estándar, como se mencionó, el ritmo y la metodología de hacer el trabajo es impuesto por el operario actualmente. La finalidad es poder ayudar en los cálculos y así estimar la forma adecuada para procesar el figurado en esta área. Para ello se diseñó el programa que permite el cálculo de los cortes, ya que su incidencia es alta y las ordenes considerablemente grandes. Por ende, el operario estaba expuesto a cometer errores en los cálculos y a eso se le suma la gran cantidad de tiempo que utilizaba para esto, con un nivel alto de incertidumbre y desconocimientos inicial de variables tales como: el número total de varillas a utilizar, el número de sobrantes y tiempo de producción del lote. Pero gracias al programa se puede conocer la cantidad de material a utilizar al inicio, ayudando con esto a mitigar la demora en la consecución de los rollos de varillas o en las faltantes y sobrante de insumos. El conocimiento predeterminado del tiempo de producción del lote, ayuda a la correcta planificación y vigilancia de los pasos a seguir, y así poder estimar que tan preparada está el área para responder a las órdenes de producción. Como ya he insistido en varias oportunidades, la inclusión de métodos clásicos, como lo es la heurística de Johnson, ayuda a la disminución del tiempo ocio y el tiempo de terminación; factores que corroborarían a una entrega pronta de los lotes de producción. Los beneficios obtenidos al momento de la aplicación de este diseño serian:

- Conocimiento predeterminado de: número de varillas estándar a utilizar como insumo.
- Tiempo de terminación de los lotes de producción en cada proceso.
- Número de cortes que no generarían sobrantes.
- Número de cortes que generarían sobrantes con sus respectivas longitudes.
- Mayor control y vigilancia en los procesos que interviene en esta área.
- Planificación y programación de la producción de una manera adecuada.
- Disminución de tiempos ocios en el área.
- Disminución de los tiempos de terminación de las órdenes de producción.

Alcances

El alcance estimado con este modelo es la correcta planificación y organización de las remisiones, logrando con ello una adecuada asignación de tiempo, mano de obra y recursos. Además, el cálculo predeterminado de los criterios anteriores ayudará a una mayor evaluación de las respectivas alternativas para así mejorar los tiempos de entregas de los lotes de producción y poder satisfacer a los clientes con respecto a las fechas de entregas, algo de vital importancia, al momento de programar la producción.

Con este modelo se pretende proponer una serie de pasos confiables, que permitan asignar las tareas a las respectivas células de trabajos, y así obtener un mayor control con el tiempo de terminación de los lotes a fabricar.

Aun así, queda abierto algunas brechas para continuar la investigación, tales como el establecimiento de días específicos para la liberación de órdenes, y establecer fechas de cortes por medio de cálculos predeterminados que facilite la planificación del tiempo necesario, recurso humano y materia prima para la elaboración de las remisiones en cola. También se es pertinente sincronizar el modelo propuesto en este trabajo de investigación con la gestión logística de la empresa, ya que como se mencionó anteriormente: como valor agregado él envió es gratis y actualmente cuenta con una flota de siete camiones con diferentes capacidades de cargas, y diferentes destinos, ya que las ordenes pueden que sean remitidas por clientes locales y/u otros municipios del departamento de Sucre, aun así también se es necesario caracterizar y priorizar a los clientes, y así establecer rutas y tiempos de despachos certeros y confiables para así mejorar los tiempos de entregas.

Recomendaciones

Como se pudo derrotar en la presente investigación en la empresa de Cronceaceros S.A.S, donde persistía el parcial cumplimiento en los tiempos de entrega de los lotes a fabricar en la planta, debido a la errónea programación de tareas y recursos en las respectivas células de trabajo. Por ende, se recomienda tomar este modelo como referencia y patrón para la asignación de las tareas en los puestos de trabajo, ya que como hemos validado y comprobado: se refleja una notable disminución en los tiempos de procesamiento, trayendo consigo una disminución en las fechas de entregas; contribuyendo a la entrega oportuna de las remisiones emitidas por los clientes.

Considerando la favorabilidad de contar con este modelo ajustado a las restricciones y ambientes de producción encontrados en la empresa de estudio, permitiéndoles la obtención de un modelo estructurado y planificado, de acuerdo a reglas factibles de programación, que optimicen los tiempos de procesado y mejore el servicio a los clientes. Además, se recomienda adoptar la restructuración propuesta en esta investigación al área de figuración estándar: ya que, se podrá contar con pasos confiables para la elaboración y programación de los trabajos, debido a que los procesos, se es requerido en un alto grado del trabajo de los operarios.

Como tercera recomendación, precisamos la importancia de que el área de figuración asistido por computador, empleen el modelo de programación de tareas propuesto en esta investigación: el algoritmo genético ajustado al ambiente de trabajos en máquinas en paralelo idénticas, trayendo a colación la importancia de respetar el orden de asignación de tareas propuestos por el algoritmo, para así poder apreciar su funcionalidad y eficiencia.

Y como última recomendación, Se ofrece con la ayuda de las reglas de despacho propuestas en esta investigación: la organización y planificación de las remisiones en cola, ayudando a la gestión y asignación de tiempo y mano de obra para el cumplimiento de los índices de entrega oportuna de las remisiones, gracias al cálculo predeterminado de los criterios evaluativos resaltados en esta investigación y así tener mayor control de la producción.

Conclusión

Con esta investigación, se puede concluir que: la forma de programar y asignar las tareas en los respectivos puestos de trabajo, afecta de una manera directa el tiempo de procesamiento, por tal motivo, aquellas empresas que necesiten de buenos índices en las fechas de entrega de las órdenes de los clientes, dependerán vitalmente del tiempo de procesado de los lotes emitidos por ellos. Por ende, se verán en la obligación de revisar y validar el modelo actual con qué programa los trabajos, y es aquí donde queremos hacer énfasis en la empresa tomada para el estudio de esta rama de la ingeniería, de la cual se observó el parcial cumplimiento de los tiempos de entrega; debido a su ineficiente forma de programar los trabajos en las respectivas células de trabajo, sumándole a ello problemas como: ineficiencia en los procesos no automatizados y falta de planificación de las remisiones en cola.

Aun así, se puede deducir que con el presente modelo propuesto, ajustado a las restricciones y ambiente de producción de la empresa tomada para esta investigación, se logra apreciar una notable disminución en los tiempos de producción; gracias al diseño de un algoritmo genético ajustado a la asignación de tareas en un ambiente de máquinas en paralelo idénticas, con la función objetivo de minimizar el tiempo de terminación máximo, y la utilización de heurísticas para la programación en los respectivos puestos de trabajo, tales como la heurística de Johnson: que pretende reducir los tiempos muertos u ocio, y la implementación de reglas de despacho: para la planificación de las remisiones en cola, todo ello apuntando a una disminución en los tiempos de entrega de los lotes a fabricar, y una mayor planificación, organización y control de la producción, acompañado de heurísticas diseñadas en esta investigación: que permita aumentar la eficiencia y eficacia en los procesos no automatizados en Concreaceros S.A.S. Con la finalidad de entregar a tiempo las remisiones u órdenes de los clientes; aumentando con ello el índice de satisfacción.

Referencias

Almeida Rodríguez, F. C., & Melián Batista, M. B. (2007). Algoritmos genéticos multimodales: Un estudio sobre la parametrización del método clearing aplicado al problema "job shop". *Actas del V Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados* , 851.

Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J., & Pazos, A. (2010). Introducción a los Algoritmos Genéticos y la Programación Genética. *Universidade da Coruña, Servizo de Publicacións*, 16.

CHASE, R., JACOBS, R., & AQUINALO, N. (2009). *Administración de operaciones producción y cadena de suministros.* McGRAW HILL.

Cruz Chávez, M. A., Frausto Solís, J., & Juárez, D. (2009). Un algoritmo de satisfactibilidad para el problema de Job Shop scheduling. Mexico.

Garey, M., & Johnson, D. (1979). *Computers and intractability a guide to the theory of np-completeness.* New york: W. H. Freeman.

Groover, M. (1997). *Fundamentos de manufacura moderna.* Pearson.

Holland, J. (1975). *Adaptación en sistemas naturales y artificiales.*

Hwan Kim, K., & Park, Y.-M. (2014). A crane scheduling method for port container terminals. *European Journal of Operational Research- ELSEIVER*, 752–768.

JOHNSON, S. (5 de Mayo de 1953). Optimal two and three-stage production schedules with setup times included. *The rand corporation*.

López Vargas, J. C. (2013). Metodología de programación en un flow shop híbrido flexible con el uso de algoritmos genéticos para reducir el makespan. Aplicacion en la industria textil. Manizales, Colombia: Universidad Nacional de Colombia .

Lopez, J., Giraldo, J., & Arango, J. (2015). Reducción del Tiempo de Terminación en la Programación de la Producción de una Línea del Flujo Híbrida Flexible (HFS). *Informacion Tecnológica*, 157-172.

MIRLEDY TORO, E., RESTREPO, Y., & GRANADA, M. (2006). ALGORITMO GENETICO MODIFICADO APLICADO AL PROBLEMA DE SECUENCIAMIENTO DE TAREAS EN SISTEMAS DE PRODUCCION LINEAL – FLOW SHOP. *Scientia et Technica Año XII*, 285-290.

Montoya Torres, J., Paternina Arboleda, C., & Frein, Y. (2002). Minimización del tiempo total de flujo de tareas en una sola máquina: Estado del arte. *Ingeniería & Desarrollo. Universidad del Norte.* , 118-12.

Park, Y.-M., & Hwan Kim, K. (2003). A crane scheduling method for port container terminals. *OR SPECTRUM*, 1-23.

Pérez Pérez, E., Pérez Castillo, I., & Jiménez Bahri, M. (2014). Algoritmo genético para secuenciación de pedidos en taller de mecanizado con máquinas en paralelo, recirculación y tiempos de preparación. *Ingeniería Industrial. Actualidad y Nuevas Tendencias* , 38-53.

Salazar - Hornig, E., & Medica -S, J. (2013). Minimización del *Makespan*en máquinas paralelas idénticas con tiempos de preparación dependientes de la secuencia utilizando algoritmo genético. *Ingenieria investigacion y tecnologia*, 43-51.

Sipper, D., & Bulfin, R. L. (1998). *Planeación y control de la producción.* McGRAW-HILL.

Velez Gallego , M. C., Castro Zuluaga, A. C., & Maya Toro, J. (2003). Algoritmo de búsqueda aleatoria para la programación de la producción en un taller de fabricación . *Revista universidad EAFIT*, 76-86.

Vicentini , F., & Puddu, S. (2003). *Algoritmos heurísticos y el problema de job shop scheduling.* Buenos Aires: Facultad de Ciencias Exactas y Naturales de la Universidad de Buenos Aires.

Anexos.

Código fuente en Matlab para la creación de la población

```
function [Poblacion_generada,tp]=poblacion_2maquina_paralelo(tp)
```

```
%% permite crear una población de las posibles secuencias de trabajos en dos maquina en paralelo.
clear  all
clc
tic
datos=xlsread ('datos','hoja1','A1:B2000'); % Tabla de Excel donde se encuentra el numero de trabajo con su respectivo tiempo de procesamiento.
n=length(datos);% número de trabajos.
tp=input('ingrese el tamaño de la poblacion: ');
a= zeros(tp,n);% matriz separadora de la maquina 1
b=zeros(tp,n); % matriz separadora de la maquina 2
%% formacion de la matriz a, que corresponde a la secuencia de operaciones de la maquina 1.
for i=1:tp %creación del cromosoma
   v_n_a=0:1:n; % vector de las posibles tareas asignar, teniendo en cuenta que el cero me sirve de guía para parar en esa máquina (1) y seguir con la siguiente (2)
   for j=1:n %creación del gen
      v_n_a;
      p_i=1/length(v_n_a);
      p_a= p_i:p_i:1;% probabilidad correspondiente
      gen_ale_a=rand(); %generador de numero aleatorio
      for e= 1:length(p_a) % evaluación del número aleatorio para cada probabilidad
         if gen_ale_a <=p_a(e)
            m=v_n_a(e); %Selección de la tarea acertada
            v_n_a(e)=[]; %eliminación de la tarea asignada
            break
         end
      end
      a(i,j)=m;
```

```
        if a(i,j)==0
%% Si se cumple el condicional, se iniciará la asignación de trabajos en la segunda
maquina
            for l=1:length(v_n_a) % asignación probabilística
                v_n_a;
                p_j=1/length(v_n_a);
                p_b=p_j:p_j:1;
                gen_ale_b=rand();
                for g=1:length(p_b)
                    if gen_ale_b <= p_b(g)
                        q=v_n_a(g);
                        v_n_a(g)=[];
                        break
                    end
                end
                b(i,l)=q;
            end


        end
    end
end
Secuencias_dos_maquinas=[a,b];% Población creada.
%% Evaluación de la población de manera individual.
n;
cmax_tp=zeros(1,tp)'; % separador de memoria para registrar los tiempo de terminacion
de cada individuo
for k =1:tp % Recorre todas las filas (población creada)
    part_1=zeros(1,n); % Separador de memoria para la maquina 1
    part_2=zeros(1,n); % Separador de memoria para la maquina 2
```

```
    for x=1:n % Extraccion de los correspondientes tiempos de procesamientos de los trabajos asignados en la maquina 1.
        k_x=Secuencias_dos_maquinas(k,x);
        if k_x==0 % si no hay ningún trabajo asignado que me salga de esa maquina y se dirija a la siguiente
            break
        end
        part_1(x)=datos(k_x,2); % almacenador
    end
    for y=n+1:n*2 % Extracción de los correspondientes tiempos de procesamientos de los trabajos asignados en la maquina 1.
        k_y=Secuencias_dos_maquinas(k,y);
        if k_y==0 % si no hay más trabajos en la segunda máquina, que me salga del ciclo.
            break
        end
        part_2(y)=datos(k_y,2); % almacenador
    end
 part_a=cumsum(part_1); % El acumulado de la maquina 1
 part_b=cumsum(part_2); % El acumulado de la maquina 2
 max_a=max(part_a); % Tiempo de terminación de la maquina 1
 max_b=max(part_b); % Tiempo de terminación de la maquina 2
 cmax_tp(k)=max(max_a,max_b); % Makespancorrespondiente al individuo.
end
Poblacion_generada=[Secuencias_dos_maquinas,cmax_tp]; % la salida del este script, ya que nos arroja los cromosomas creados  con su respectiva evaluación.

toc
end
```

Código fuente del operador genético en Matlab

```
function [Poblacion_temporal]=Operadores_geneticos(it,p_c,p_m)
clear all
clc
%% Fase II Definición de los operadores genéticos.
%% 2.1 Selección, por torneo= Selecciona dos padres al azar y escoge el mejor de ellos según la función objetivo (Cmax)
% el padre escogido pasa a la matriz de posibles padres a tener hijos y el
% otro se rechaza, pero ambos permanecen en la poblacion_generada, SIN
% eliminarse...
tic
[Poblacion_generada]=poblacion_2maquina_paralelo;
[tp,n]=size(Poblacion_generada);
it= input('ingrese el numero de iteraciones: ');
p_c= input('ingrese la probabilidad de cruzamiento: ');
p_m= input('ingrese la probabilidad de mutacion: ');
Posible_padre=zeros(2,n);% separador de la memoria para los padres aptos
h_1=zeros(1,n);
h_1=zeros(1,n);
M_H=zeros(2,n);
Poblacion_temporal=zeros(it,n);
for k=1:it
   %%selección por torneo
   for a=1:1:2 % el proceso de selección se realiza dos veces por iteración, para poder escoger los dos posibles padres
      individuo_1=Poblacion_generada(floor(1+rand*tp),:); % se escoge al azar al primer
```

```
individuo
      individuo_2=Poblacion_generada(floor(1+rand*tp),:); % se escoge al azar al
segundo individuo
      if individuo_1(n)>individuo_2(n) % se evalúa cuál de ellos es el mejor según la
función objetivo, para escoger el mejor individuo
         Posible_padre(a,:)= individuo_2;
      else
         Posible_padre(a,:)= individuo_1;
      end
      Posible_padre; % los seleccionados se guardan en esta matriz.
   end
   Posible_padre;
   %% Cruzamiento
   num_ale= rand;
   q=n-1;
   if num_ale <= p_c
      if Posible_padre(1,1)~=0 && Posible_padre(2,1)~=0 &&
Posible_padre(1,q/2+1)~=0 && Posible_padre(2,q/2+1)~=0 % evaluar si los padres
entran en caso 1.
         h_1=[Posible_padre(1,1), Posible_padre(2,q/2+2:q), Posible_padre(1,q/2+1),
Posible_padre(2,2:q/2)];
         h_2=[Posible_padre(2,1), Posible_padre(1,q/2+2:q), Posible_padre(2,q/2+1),
Posible_padre(1,2:q/2)];
elseif Posible_padre(1,1)==0 || Posible_padre(2,1)==0 || Posible_padre(1,q/2+1)==0 ||
Posible_padre(2,q/2+1)==0 % evaluar si los padres entran en caso 2 y 3.
      if Posible_padre(1,1)==0
         Posible_padre(1,1)=Posible_padre(1,q/2+1);
         Posible_padre(1,q)=0;
      end
```

```
    if Posible_padre(2,1)==0
      Posible_padre(2,1)=Posible_padre(2,q/2+1);
      Posible_padre(2,q)=0;
    end
    if Posible_padre(1,q/2+1)==0
       Posible_padre(1,q/2+1)= Posible_padre(1,1);
       Posible_padre(1,q/2)=0;
    end
    if Posible_padre(2,q/2+1)==0
      Posible_padre(2,q/2+1)=Posible_padre(2,1);
      Posible_padre(2,q/2)=0;
    end
    Posible_padre;
    h_1=[Posible_padre(1,1), Posible_padre(2,q/2+2:q), Posible_padre(1,q/2+1),
Posible_padre(2,2:q/2)];
    h_2=[Posible_padre(2,1), Posible_padre(1,q/2+2:q), Posible_padre(2,q/2+1),
Posible_padre(1,2:q/2)];

    end

  else % sino cae en la probabilidad de cruce.
     continue
  end
  % Evaluación de los hijos creados, para poder eliminar a los repetidos
  % e incluir los trabajos que hacen falta por asignar.
  h_1;
  h_2;
  datos=xlsread ('datos','hoja1','A1:B2000'); % Tabla de Excel donde se encuentra el
numero de trabajo con su respectivo tiempo de procesamiento.
```

```
N=length(datos);% número de trabajos.
M_H=[h_1;h_2]; % Matriz que contiene a los hijos creados.
for b= 1:2
   hijo_evaluado= M_H(b,:); % se escoge el hijo a evaluar
   v_unicos= unique(hijo_evaluado); % se sacan los alelos únicos del respectivo hijo
   N= 0:1:length(datos); % se crea una vector del cero hasta el número de trabajos
   for c=1:length(v_unicos) % se utiliza para eliminar los alelos ya asignados en el
hijo, y asi obtener a los alelos faltantes
      for d= 1:length(N)
         if v_unicos(c)== N(d)
            N(d)=[];
            break
         end
      end
   end
   v_f=N; % vector que contiene los alelos que hacen falta por asignar en el hijo
   for e=1:length( hijo_evaluado) % Se evalúa los alelos repetidos en el hijo, para
quitarlo y remplazarlo por los que se encuentran en la variable v_f
      evaluador= hijo_evaluado(e);
      if evaluador == 0
         continue
      end
      for f=1:length( hijo_evaluado)
         if e==f
            continue
         end
       if evaluador== hijo_evaluado(f)
          hijo_evaluado(f)=v_f(1);
          v_f(1)=[];
```

```
            break
        end
        end
    end
    M_H(b,:)=hijo_evaluado;
end
% Evaluar según la F.o los hijos creados y así tomar al mejor.
M_H;
Ceros=[0 0];
Datos=[Ceros;datos]; % incluyendo el trabajo 0 con su respectivo tp =0
Cmax=zeros(2,1); % separador de memoria, para el almacenamiento del Makespan
T_P=zeros(2,q); % separador de memoria, para el almacenamiento de los tiempo de procesado
for s=1:2
    q; % Numero de trabajos multiplicados por dos
    z=q/2; % Numero de trabajos

    for t= 1:q % Se saca los respectivos trabajos para así obtener su respectivo tiempo de procesado.
        indice=M_H(s,t);
        if indice ==0
            continue
        else
        T_P(s,t)= datos(indice,2);
        end

    end
    parte_1= cumsum(T_P(s,1:z));
    parte_2= cumsum(T_P(s,z+1:q));
```

```
        max_1=max(parte_1); % tiempo máximo de la maquina numero 1
        max_2=max(parte_2); % tiempo máximo de la maquina numero 2
        Cmax(s,1)= max(max_1,max_2); % se escoge el Cmax de las dos máquinas en
paralelo


    end
    M_H;
    Cmax;
    Hijos_creados=[M_H,Cmax];
    % Se revisa quien tiene el menor Cmax de los hijos creados para que
    % entren en la sgte fase (mutación).
    if Cmax(1)> Cmax(2)
        h_eleg_mut=[M_H(2,:),Cmax(2)];
    elseif Cmax(2)> Cmax(1)
        h_eleg_mut=[M_H(1,:),Cmax(1)];
    else
        h_eleg_mut=[M_H(1,:),Cmax(1)];
    end

    %% Mutación
    aleatorio_num=rand;
    h_eleg_mut; % hijo que posiblemente será mutado
    h_e_m=h_eleg_mut; % me toco hacer esto para no perder el cromosoma hijo antes de
ser mutado.
    n_=q/2; % número de trabajos
    if aleatorio_num <=p_m
        for kk=1:100000000 % se busca el índice dentro del hijo, en donde ambas maquinas
estén asignados trabajos para su respectivo cambio(se excluye el cero=
            ale_gen= floor(1+(rand*n_));
```

```
        if h_e_m(ale_gen)~=0 & h_e_m(n_+ale_gen)~=0
            almacenador_=h_e_m(ale_gen);
            h_e_m(ale_gen)= h_e_m(ale_gen+n_);
            h_e_m(ale_gen+n_)=almacenador_;
            h_e_m(q+1)=[]; %eliminador del Makespandel hijo antes de ser mutado.
            hijo_mutado=h_e_m; % hijo mutado
            break
        else
            ale_gen= floor(1+(rand*n_)); % se genera nuevamente el numero aleatorio
        end
    end

% Evaluación según las FO del hijo antes de ser mutado vs hijo mutado.
h_eleg_mut;
hijo_mutado;
% buscar el Makespandel hijo mutado.
datos; % incluyendo el trabajo 0 con su respectivo tp =0
tp_m=zeros(1,q);
for w =1:q
    ind_m=hijo_mutado(w);
    if ind_m~=0
      tp_m(w)=datos(ind_m,2);
    else
      tp_m(w)=0;
    end
end
parte_1_m= cumsum(tp_m(1:n_));
parte_2_m= cumsum(tp_m(n_+1:q));
max_1_m=max(parte_1_m);
```

```
    max_2_m=max(parte_2_m);
    Cmax_m= max(max_1_m,max_2_m);
    hijo_mutado=[hijo_mutado,Cmax_m];
    % se escoge el mayor entre el hijo sin mutar y el mutado
    h_eleg_mut;
    hijo_mutado;
    if h_eleg_mut(n)> hijo_mutado(n)
        Poblacion_temporal (k,:)=hijo_mutado;
    elseif h_eleg_mut(n)< hijo_mutado(n)
        Poblacion_temporal (k,:)=h_eleg_mut;
    else
        Poblacion_temporal (k,:)=hijo_mutado;
    end
    else
        Poblacion_temporal (k,:)= h_eleg_mut; % si no cae el numero aleatorio en la
probabilidad de mutación
    end
end
% poder eliminar los ceros
Poblacion_temporal;
for i=1:tp
    Poblacion_temporal;
    if all(Poblacion_temporal(:,n)~=0)
        break
    end
    for j=1:length(Poblacion_temporal)
        if Poblacion_temporal(j,n)==0
            Poblacion_temporal(j,:)=[];
            break
```

```
        end
    end
end

Poblacion_temporal;% los resultados finales, al pasar por todo estos operadores, se
almacenan en esta variables.

end
```

Código fuente en Matlab del resultado final del Algoritmo genético

```
%% Se escoge el mejor según la función objetivo...
function [Maquina_1,Maquina_2,Tiempo_terminacion]= Resultado_genetico
clear all
```

```
clc
[Poblacion_temporal]=Operadores_geneticos;
[tp,n]=size(Poblacion_temporal);
m=((n-1)/2);
q=m+1;
z=n-1;

Resultado=zeros(1,n);
Maquina_1= zeros(1,m);
Maquina_2=zeros(1,m);
Poblacion_temporal;
Makespan=Poblacion_temporal(:,n);
[Cmax,indice]=min(Makespan);
Resultado=Poblacion_temporal(indice,1:n);



Maquina_1= Resultado(1:m)
Maquina_2= Resultado(q:z)
Tiempo_terminacion= Resultado(n)
end
```

Código fuente de la heurística unificación sin sobrantes en Matlab

```
%% Realiza los Cortes posibles sin dejar longitudes sobrantes o punta alguna.
clear all
```

```
clc
tic
Insumo=input('Ingrese el tamaño de la varilla estandar a utilizar: ');
Datos=xlsread('Datos','Hoja1','A1:C10000');
Datos_2=xlsread('Datos','Hoja1','A1:C10000');
n=1;  v_unidad=0;    Resultado_1=[];       % Unicos
mm=1; v_cantidad=0;  Resultado_Cantidad=[]; % Cantidad
l=1;  v_parejas=0;   Resultado_2=[];       % Parejas
m=1;  v_trios=0;     Resultado_3=[];       % Trio
p=1;  v_cuarteto=0;  Resultado_4=[];       % Cuarteto
q=1;  v_quinteto=0;  Resultado_5=[];       % Quinteto
s=1;  v_sexteto=0;   Resultado_6=[];       % Sexteto
Sum_cortes_realizados_cantidad=0; % En el momento de sacar el número de cortes
realizados en esta sección.
%% Unidad
for ii=1:length(Datos)
   if Datos(ii,2)==Insumo
      % almacenador de resultado en una estructura.
      Resultado_1(n). Trabajo_1=Datos_2(ii,1);
      Resultado_1(n). Tipo_Corte=Datos_2(ii,2);
      Resultado_1(n). Cantidad= Datos_2(ii,3);
      Resultado_1(n). Sobrante=0;

      Resultado_1(n). Numero_varillas_utilizadas=Datos_2(ii,3);
      v_unidad=sum([Resultado_1.Numero_varillas_utilizadas]); % sumatoria de las
varillas utilizadas en esta sección
      n=n+1;
      Datos(ii,2)=0;
      Datos(ii,3)=0;
```

```
    end
end
n=n-1; % número de trabajos terminados.
%% Parejas
for j=1:length(Datos)
    if Datos(j,3)==0
        continue

    end
    Indice_1=Datos(j,2);
    for i=1:length (Datos)
        if j==i
            continue
        end
        if Datos(i,3)==0
            continue
        end
        Indice_2=Datos(i,2);
        if Indice_1+Indice_2==Insumo && Datos(j,3)>0 && Datos(i,3)>0
            v=[Datos(j,3),Datos(i,3)];
            Cantidad_menor= min(Datos(j,3),Datos(i,3));
            Datos(j,3)=Datos(j,3)-Cantidad_menor;
            Datos(i,3)=Datos(i,3)-Cantidad_menor;

            % almacenador de resultado en una estructura.
            Resultado_2(l). Trabajo_1=Datos(j,1);
            Resultado_2(l). Tipo_Corte=Datos(j,2);
            Resultado_2(l). Cantidad= v(1);
            Resultado_2(l). Sobrante=Datos(j,3);
```

```
        Resultado_2(l). Trabajo_2=Datos(i,1);
        Resultado_2(l). Tipo_Corte_2=Datos(i,2);
        Resultado_2(l). Cantidad_2= v(2);
        Resultado_2(l). Sobrante_2=Datos(i,3);

        Resultado_2(l). Numero_varillas_utilizadas=Cantidad_menor;
        l=l+1;
        v_parejas=sum([Resultado_2. Numero_varillas_utilizadas]); % sumatoria de las
varillas utilizadas en esta sección
        if Datos(j,3)==0
          Datos(j,2)=0;
        end
        if Datos (i,3)==0
          Datos(i,2)=0;

        end
        Datos;
      end
   end
end
l=l-1; % Número de Parejas formadas
%% Tríos
for b= 1:length(Datos)
   if Datos(b,3)~=0
     Indice_1A=Datos(b,2);
   else
     continue
   end
```

```
    for c=1:length(Datos)
        if b==c
            continue
        elseif Datos(c,3)~=0
             Indice_1B=Datos(c,2);
        else
            continue
        end
        Indice_1B=Datos(c,2);
        for d=1:length(Datos)
            if b==d
                continue
            elseif c==d
                continue
            elseif Datos (d,3)~=0
                Indice_1C=Datos(d,2);
            else
                continue
            end

            if  Indice_1A+Indice_1B+Indice_1C==Insumo & Datos(b,3)>0 & Datos(c,3)>0
& Datos(d,3)>0
                v=[Datos(b,3),Datos(c,3),Datos(d,3)];
                cantidad_menor2=min(v);
                Datos(b,3)=Datos(b,3)-cantidad_menor2;
                Datos(c,3)=Datos(c,3)-cantidad_menor2;
                Datos(d,3)=Datos(d,3)-cantidad_menor2;
```

```
Resultado_3(m). Trabajo1= Datos(b,1);
Resultado_3(m). Tipo_Corte1= Datos(b,2);
Resultado_3(m). Cantidad1= v(1);
Resultado_3(m). Sobrante1= Datos(b,3);

Resultado_3(m). Trabajo2= Datos(c,1);
Resultado_3(m). Tipo_Corte2= Datos(c,2);
Resultado_3(m). Cantidad2= v(2);
Resultado_3(m). Sobrante2= Datos(c,3);

Resultado_3(m). Trabajo3= Datos(d,1);
Resultado_3(m). Tipo_Corte3= Datos(d,2);
Resultado_3(m). Cantidad3= v(3);
Resultado_3(m). Sobrante3= Datos(d,3);

Resultado_3(m). Numero_varillas_utilizadas= cantidad_menor2;

if Datos (b,3)==0
   Datos(b,2)=0;
end
if Datos(c,3)==0
   Datos(c,2)=0;
end
if Datos(d,3)==0
   Datos(d,2)=0;
end
v_trios=sum([Resultado_3. Numero_varillas_utilizadas]); % sumatoria de las varillas utilizadas en esta sección
m=m+1;
```

```
            continue
        end

    end

  end

end
m=m-1; % número de tríos formados
%% Cuartetos
for b_1= 1:length(Datos)
  if Datos(b_1,3)~=0
    Indice_A=Datos(b_1,2);
  else
    continue
  end

  for c_1=1:length(Datos)
    if b_1==c_1
      continue
    elseif Datos(c_1,3)~=0
       Indice_B=Datos(c_1,2);
    else
      continue
    end
    for d_1=1:length(Datos)
      if b_1==d_1
        continue
      elseif c_1==d_1
```

```
        continue
    elseif Datos (d_1,3)~=0
        Indice_C=Datos(d_1,2);
    else
        continue
    end
    for f_1=1:length(Datos);
        if b_1==f_1
            continue
        elseif c_1==f_1
            continue
        elseif d_1==f_1
            continue
        elseif Datos(f_1,3)~=0
            Indice_D=Datos(f_1,2);
        else
            continue
        end

        if  Indice_A+Indice_B+Indice_C+Indice_D==Insumo && Datos(b_1,3)>0 &&
Datos(c_1,3)>0 && Datos(d_1,3)>0 && Datos(f_1,3)>0

        v=[Datos(b_1,3),Datos(c_1,3),Datos(d_1,3),Datos(f_1,3)];
        cantidad_menor2=min(v);
        Datos(b_1,3)=Datos(b_1,3)-cantidad_menor2;
        Datos(c_1,3)=Datos(c_1,3)-cantidad_menor2;
        Datos(d_1,3)=Datos(d_1,3)-cantidad_menor2;
        Datos(f_1,3)=Datos(f_1,3)-cantidad_menor2;
```

```
Resultado_4(p). Trabajo1= Datos(b_1,1);
Resultado_4(p). Tipo_Corte1= Datos(b_1,2);
Resultado_4(p). Cantidad1= v(1);
Resultado_4(p). Sobrante1= Datos(b_1,3);

Resultado_4(p). Trabajo2= Datos(c_1,1);
Resultado_4(p). Tipo_Corte2= Datos(c_1,2);
Resultado_4(p). Cantidad2= v(2);
Resultado_4(p). Sobrante2= Datos(c_1,3);

Resultado_4(p). Trabajo3= Datos(d_1,1);
Resultado_4(p). Tipo_Corte3= Datos(d_1,2);
Resultado_4(p). Cantidad3= v(3);
Resultado_4(p). Sobrante3= Datos(d_1,3);

Resultado_4(p). Trabajo4= Datos(f_1,1);
Resultado_4(p). Tipo_Corte4= Datos(f_1,2);
Resultado_4(p). Cantidad4= v(4);
Resultado_4(p). Sobrante4= Datos(f_1,3);

Resultado_4(p). Numero_varillas_utilizadas= cantidad_menor2;

if Datos (b_1,3)==0
   Datos(b_1,2)=0;
elseif Datos(c_1,3)==0
    Datos(c_1,2)=0;
elseif Datos(d_1,3)==0
    Datos(d_1,2)=0;
elseif Datos(f_1,3)==0
```

```
                Datos(f_1,2)=0;
            end
            p=p+1;
            v_cuarteto=sum([Resultado_4. Numero_varillas_utilizadas]);
           continue
            end
        end

    end

  end

end
p=p-1;% Numero de cuartetos armados
%% Quintetos
for b_2= 1:length(Datos)
  if Datos(b_2,3)~=0
    Indice_A=Datos(b_2,2);
  else
    continue
  end

  for c_2=1:length(Datos)
    if b_2==c_2
      continue
    elseif Datos(c_2,3)~=0
      Indice_B=Datos(c_2,2);
    else
      continue
```

```
end
for d_2=1:length(Datos)
   if b_2==d_2
      continue
   elseif c_2==d_2
      continue
   elseif Datos (d_2,3)~=0
      Indice_C=Datos(d_2,2);
   else
      continue
   end
   for f_2=1:length(Datos);
      if b_2==f_2
         continue
      elseif c_2==f_2
         continue
      elseif d_2==f_2
         continue
      elseif Datos(f_2,3)~=0
         Indice_D=Datos(f_2,2);
      else
         continue
      end

      for e_2=1:length(Datos)
         if e_2==b_2
            continue
         elseif c_2==e_2
            continue
```

```
elseif d_2==e_2
   continue
elseif f_2==e_2
   continue
elseif Datos(e_2,3)~=0
   Indice_E=Datos(e_2,2);
else
   continue
end

if  Indice_A+Indice_B+Indice_C+Indice_D+Indice_E==Insumo &&
Datos(b_2,3)>0 && Datos(c_2,3)>0 && Datos(d_2,3)>0 && Datos(f_2,3)>0 &&
Datos(e_2,3)>0

   v=[Datos(b_2,3),Datos(c_2,3),Datos(d_2,3),Datos(f_2,3),Datos(e_2,3)];
   cantidad_menor2=min(v);
   Datos(b_2,3)=Datos(b_2,3)-cantidad_menor2;
   Datos(c_2,3)=Datos(c_2,3)-cantidad_menor2;
   Datos(d_2,3)=Datos(d_2,3)-cantidad_menor2;
   Datos(f_2,3)=Datos(f_2,3)-cantidad_menor2;
   Datos(e_2,3)=Datos(e_2,3)-cantidad_menor2;

   Resultado_5(q). Trabajo1= Datos(b_2,1);
   Resultado_5(q). Tipo_Corte1= Datos(b_2,2);
   Resultado_5(q). Cantidad1= v(1);
   Resultado_5(q). Sobrante1= Datos(b_2,3);

   Resultado_5(q). Trabajo2= Datos(c_2,1);
   Resultado_5(q). Tipo_Corte2= Datos(c_2,2);
```

```
Resultado_5(q). Cantidad2= v(2);
Resultado_5(q). Sobrante2= Datos(c_2,3);

Resultado_5(q). Trabajo3= Datos(d_2,1);
Resultado_5(q). Tipo_Corte3= Datos(d_2,2);
Resultado_5(q). Cantidad3= v(3);
Resultado_5(q). Sobrante3= Datos(d_2,3);

Resultado_5(q). Trabajo4= Datos(f_2,1);
Resultado_5(q). Tipo_Corte4= Datos(f_2,2);
Resultado_5(q). Cantidad4= v(4);
Resultado_5(q). Sobrante4= Datos(f_2,3);

Resultado_5(q). Trabajo5= Datos(e_2,1);
Resultado_5(q). Tipo_Corte5= Datos(e_2,2);
Resultado_5(q). Cantidad5= v(5);
Resultado_5(q). Sobrante5= Datos(e_2,3);

Resultado_5(q). Numero_varillas_utilizadas= cantidad_menor2;

if Datos (b_2,3)==0
   Datos(b_2,2)=0;
elseif Datos(c_2,3)==0
   Datos(c_2,2)=0;
elseif Datos(d_2,3)==0
   Datos(d_2,2)=0;
elseif Datos(f_2,3)==0
   Datos(f_2,2)=0;
elseif Datos (e_2,3)==0
```

```
                    Datos(e_2,2)=0
                end
                q=q+1;
                v_quinteto=sum([Resultado_5. Numero_varillas_utilizadas]); % suma el
número de varillas utilizadas en esta sección
                continue
            end
        end

    end

  end
end
q=q-1;% Numero de cuartetos armados
%% sextetos
for b_3= 1:length(Datos)
  if Datos(b_3,3)~=0
    Indice_A=Datos(b_3,2);
  else
    continue
  end

  for c_3=1:length(Datos)
    if b_3==c_3
      continue
    elseif Datos(c_3,3)~=0
      Indice_B=Datos(c_3,2);
    else
```

```
    continue
end
for d_3=1:length(Datos)
    if b_3==d_3
        continue
    elseif c_3==d_3
        continue
    elseif Datos (d_3,3)~=0
        Indice_C=Datos(d_3,2);
    else
        continue
    end
    for f_3=1:length(Datos);
        if b_3==f_3
            continue
        elseif c_3==f_3
            continue
        elseif d_3==f_3
            continue
        elseif Datos(f_3,3)~=0
            Indice_D=Datos(f_3,2);
        else
            continue
        end

        for e_3=1:length(Datos)
            if e_3==b_3
                continue
            elseif c_3==e_3
```

```
    continue
elseif d_3==e_3
    continue
elseif f_3==e_3
    continue
elseif Datos(e_3,3)~=0
    Indice_E=Datos(e_3,2);
else
    continue
end
for g_3=1:length(Datos)
    if g_3==b_3
        continue
    elseif g_3==c_3
        continue
    elseif g_3==d_3
        continue
    elseif g_3==f_3
        continue
    elseif g_3==e_3
        continue
    elseif Datos(g_3,3)~=0
        Indice_G=Datos(g_3,2);
    else
        continue
    end

    if  Indice_A+Indice_B+Indice_C+Indice_D+Indice_E+
Indice_G==Insumo && Datos(b_3,3)>0 && Datos(c_3,3)>0 && Datos(d_3,3)>0 &&
```

```
Datos(f_3,3)>0 && Datos(e_3,3)>0&& Datos(g_3,3)>0


v=[Datos(b_3,3),Datos(c_3,3),Datos(d_3,3),Datos(f_3,3),Datos(e_3,3),Datos(g_3,3)];
                cantidad_menor2=min(v);
                Datos(b_3,3)=Datos(b_3,3)-cantidad_menor2;
                Datos(c_3,3)=Datos(c_3,3)-cantidad_menor2;
                Datos(d_3,3)=Datos(d_3,3)-cantidad_menor2;
                Datos(f_3,3)=Datos(f_3,3)-cantidad_menor2;
                Datos(e_3,3)=Datos(e_3,3)-cantidad_menor2;
                Datos(g_3,3)=Datos(g_3,3)-cantidad_menor2;

                Resultado_6(s). Trabajo1= Datos(b_3,1);
                Resultado_6(s). Tipo_Corte1= Datos(b_3,2);
                Resultado_6(s). Cantidad1= v(1);
                Resultado_6(s). Sobrante1= Datos(b_3,3);

                Resultado_6(s). Trabajo2= Datos(c_3,1);
                Resultado_6(s). Tipo_Corte2= Datos(c_3,2);
                Resultado_6(s). Cantidad2= v(2);
                Resultado_6(s). Sobrante2= Datos(c_3,3);

                Resultado_6(s). Trabajo3= Datos(d_3,1);
                Resultado_6(s). Tipo_Corte3= Datos(d_3,2);
                Resultado_6(s). Cantidad3= v(3);
                Resultado_6(s). Sobrante3= Datos(d_3,3);

                Resultado_6(s). Trabajo4= Datos(f_3,1);
                Resultado_6(s). Tipo_Corte4= Datos(f_3,2);
```

```
Resultado_6(s). Cantidad4= v(4);
Resultado_6(s). Sobrante4= Datos(f_3,3);

Resultado_6(s). Trabajo5= Datos(e_3,1);
Resultado_6(s). Tipo_Corte5= Datos(e_3,2);
Resultado_6(s). Cantidad5= v(5);
Resultado_6(s). Sobrante5= Datos(e_3,3);

Resultado_6(s). Trabajo6= Datos(g_3,1);
Resultado_6(s). Tipo_Corte6= Datos(g_3,2);
Resultado_6(s). Cantidad6= v(6);
Resultado_6(s). Sobrante6= Datos(g_3,3);

Resultado_6(s). Numero_varillas_utilizadas= cantidad_menor2

if Datos (b_3,3)==0
   Datos(b_3,2)=0;
elseif Datos(c_3,3)==0
   Datos(c_3,2)=0;
elseif Datos(d_3,3)==0
   Datos(d_3,2)=0;
elseif Datos(f_3,3)==0
   Datos(f_3,2)=0;
elseif Datos (e_3,3)==0
   Datos(e_3,2)=0;
elseif Datos(g_3,3)==0
   Datos(g_3,2)=0;
end
s=s+1;
```

```
                    v_sexteto=sum([Resultado_6. Numero_varillas_utilizadas]); % Suma el
número de varillas de esta sección.
                        continue
                    end
                end

            end

        end
    end
end
s=s-1;% Numero de cuartetos armados
%% Según su cantidad.
for kk=1:length(Datos)
    y=0;
    if Datos(kk,2)~=0
        if rem(Insumo,Datos(kk,2))==0
            Coeficiente_division=Insumo/Datos(kk,2);
            if Datos(kk,3)>=Coeficiente_division
              Comparador=Datos(kk,3);
               for kkk=1:Datos(kk,3)
                  if Datos(kk,3)<Coeficiente_division
                     break
                  end
                  Diferencia= Datos(kk,3)-Coeficiente_division;
                  y=y+1;
                  if Datos(kk,3)<Coeficiente_division
                     break
```

```
                else
                    Datos(kk,3)=Datos(kk,3)- Coeficiente_division;
                end
            end
            Datos(kk,3)=Comparador-(y*Coeficiente_division);
            Resultado_Cantidad(mm). Trabajo= Datos(kk,1);
            Resultado_Cantidad(mm). Tipo__Corte= Datos(kk,2);
            Resultado_Cantidad(mm). Cantidad= Comparador;
            Resultado_Cantidad(mm). Sobrante= Datos(kk,3);
            Resultado_Cantidad(mm). Cantidad_cortes= (Coeficiente_division-1)*y; %
indica el número de cortes realizados
            Resultado_Cantidad(mm). Numero_varillas_utilizadas= y;
            v_cantidad=sum([Resultado_Cantidad. Numero_varillas_utilizadas]); %
número de varillas utilizadas en esta sección.
            Sum_cortes_realizados_cantidad=sum([Resultado_Cantidad(mm).
Cantidad_cortes]);
            mm=mm+1;
            if Datos(kk,3)==0
                Datos(kk,2)=0;
            end
        end
    end


  end
end
mm=mm-1; % Numero de Cantidades formadas
%% Resultado
% Números de trabajos terminados.
Datos;
```

```
Datos_2;
m_comparacion=zeros(length(Datos_2),3);
ff=1;
fprintf('\nTrabajos terminados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad \n');
for jj=1:length(Datos)
   if Datos(jj,2)==0
      m_comparacion(ff,1)=Datos_2(jj,1); % posición
      m_comparacion(ff,2)=Datos_2(jj,2); % Tipo de corte
      m_comparacion(ff,3)=Datos_2(jj,3); % Cantidad
      fprintf('%6d %12.2f %11d \n', m_comparacion(ff,1),
m_comparacion(ff,2),m_comparacion(ff,3));
      ff=1+ff;
   end
end
fprintf('\nNumero de trabajos terminados:\t');
fprintf('%1d\n',ff-1);

% Numero de trabajos empezados
Datos;
Datos_2;
n_comparador=zeros(length(Datos_2),4);
gg=0;
fprintf('\nTrabajos empezados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad Realizada  Cantidad Faltante  \n');
for ii=1:length(Datos)
   if Datos(ii,2)~=0
      gg;
       if Datos (ii,3)~= Datos_2(ii,3)
```

```
        gg=gg+1;
        n_comparador(gg,1)= Datos(ii,1); % N° del trabajo respectivo
        n_comparador(gg,2)= Datos(ii,2); % Tipo de corte
        n_comparador(gg,3)= Datos_2(ii,3)-Datos(ii,3); % Cantidad Realizada
        n_comparador(gg,4)= Datos(ii,3); % Cantidad Faltante
        fprintf('%6d %12.2f %16d %19d \n',n_comparador(gg,1),
n_comparador(gg,2),n_comparador(gg,3),n_comparador(gg,4));
      end
   end
end
n_comparador; %Matriz almacenadora
disp('Numero de trabajos empezados: ');
disp(gg);
disp('Sumatoria de las cantidades termiandas en los trabajos emprezados:');
disp(sum(n_comparador(:,3)));
disp('Sumatoria de las cantidades faltantes en los trabajos empezados:');
disp(sum(n_comparador(:,4)));

% Numero de trabajos NO tocados
Datos;
Datos_2;
y_comparador=zeros(length(Datos_2),3);
cc=0;
fprintf('\nTrabajos No tocados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad \n');
for yy=1:length(Datos)
   if Datos(yy,2)~=0
      cc;
       if Datos (yy,3)== Datos_2(yy,3)
```

```
            cc=cc+1;
            y_comparador(cc,1)= Datos(yy,1); % N° del trabajo respectivo
            y_comparador(cc,2)= Datos(yy,2); % Tipo de corte
            y_comparador(cc,3)= Datos(yy,3); % Cantidad sin tocar
            fprintf('%6d %12.2f %11d \n', y_comparador(cc,1),
y_comparador(cc,2),y_comparador(cc,3));
        end
    end
end
y_comparador;
disp('Numero de trabajos no tocados:')
disp(cc);
disp('Sumatoria de las cantidad de cortes no tocados:');
disp(sum(y_comparador(:,3)));

% Número total de varillas utilizadas.
Total_Varillas=sum([v_unidad,v_parejas,v_trios,v_cuarteto, v_quinteto, v_sexteto,
v_cantidad]); % Cantidad total de varillas utilizadas en este Script
fprintf('\nNumero Total de Varillas utilizadas es:\t');
fprintf('%1d\n',Total_Varillas);


% Numero de cortes totales realizados
Numero_cortes_totales=v_unidad*0 + v_parejas*1 + v_trios*2 + v_cuarteto*3 +
v_quinteto*4 + v_sexteto*5 + Sum_cortes_realizados_cantidad;
fprintf('\nNumero Total de Cortes realizados:\t');
fprintf('%1d\n',Numero_cortes_totales);

% El resultado lo almacena en Excel, ya que alimentara a la función
```

```
% siguiente.
[Estado]= xlswrite('Resultado(1)', Datos,'Hoja1','A1');
toc
```

Código fuente de la heurística unificación con sobrantes en Matlab

```
clear all
clc
Datos=xlsread('Resultado(1)','Hoja1','A1:C10000'); % Contiene los resultados del Script anterior.
Insumo=input('Ingrese el tamaño de la varilla estandar a utilizar: ');
Rango_P_S= input('ingrese el Rango de Permisibilidad superior respecto a la longitud del sobrante: '); % me indica la máxima longitud permitida.
Rango_P_I= input('ingrese el Rango de Permisibilidad Inferior respecto a la longitud del sobrante: '); % me indica la mínima longitud permitida.
Datos_2=xlsread('Datos','Hoja1','A1:C10000');
n=1;  v_unidad=0;   vS_unidad=0;    Resultado_1=[];       % Únicos.
mm=1; v_cantidad=0; vS_cantidad=0;  Resultado_Cantidad=[]; % Cantidad.
l=1;  v_parejas=0;  vS_parejas=0;   Resultado_2=[];       % Parejas.
m=1;  v_trios=0;    vS_trios=0;     Resultado_3=[];       % Trio.
p=1;  v_cuarteto=0; vS_cuarteto=0;  Resultado_4=[];       % Cuarteto.
q=1;  v_quinteto=0; vS_quinteto=0;  Resultado_5=[];       % Quinteto.
s=1;  v_sexteto=0;  vS_sexteto=0;   Resultado_6=[];       % Sexteto.
Sum_cortes_realizados_cantidad=0;                         % En el momento de sacar el número de cortes realizados en esta sección.
%% Unidad.
for ii=1:length(Datos)
   Diferencia= Insumo-Datos(ii,2);
   if Diferencia <=Rango_P_S && Diferencia >=Rango_P_I
      % almacenador de resultado en una estructura.
```

```
        Resultado_1(n). Trabajo_1=Datos(ii,1);
        Resultado_1(n). Tipo_Corte=Datos(ii,2);
        Resultado_1(n). Cantidad= Datos(ii,3);

        Resultado_1(n). Numero_varillas_utilizadas= Datos(ii,3);
        Resultado_1(n). Longitud_punta=Diferencia;
        Resultado_1(n). Cantidad_puntas_sobrantes=Datos(ii,3);
        v_unidad=sum([Resultado_1.Numero_varillas_utilizadas]); % sumatoria de las varillas utilizadas en esta sección
        vS_unidad=sum([Resultado_1.Cantidad_puntas_sobrantes]); % sumatoria de las longitudes de puntas sobrantes en esta sección
        n=n+1;
        Datos(ii,2)=0;
        Datos(ii,3)=0;
    end
end
n=n-1; % numero de trabajos terminados.
%% Parejas
for j=1:length(Datos)
    if Datos(j,3)==0
        continue
    end
    Indice_1=Datos(j,2);
    for i=1:length (Datos)
        if j==i
            continue
        end
        if Datos(i,3)==0
            continue
```

```
        end
        Indice_2=Datos(i,2);
        Diferencia=Insumo-(Indice_1+Indice_2);
        if Diferencia <=Rango_P_S && Diferencia >=Rango_P_I && Datos(j,3)>0 &&
Datos(i,3)>0
            v=[Datos(j,3),Datos(i,3)];
            Cantidad_menor= min(Datos(j,3),Datos(i,3));
            Datos(j,3)=Datos(j,3)-Cantidad_menor;
            Datos(i,3)=Datos(i,3)-Cantidad_menor;

            % almacenardor de resultado en una estructura.
            Resultado_2(l). Trabajo_1=Datos(j,1);
            Resultado_2(l). Tipo_Corte=Datos(j,2);
            Resultado_2(l). Cantidad= v(1);
            Resultado_2(l). Sobrante=Datos(j,3);

            Resultado_2(l). Trabajo_2=Datos(i,1);
            Resultado_2(l). Tipo_Corte_2=Datos(i,2);
            Resultado_2(l). Cantidad_2= v(2);
            Resultado_2(l). Sobrante_2=Datos(i,3);

            Resultado_2(l). Numero_varillas_utilizadas= Cantidad_menor;
            Resultado_2(l). Longitud_punta=Diferencia;
            Resultado_2(l). Cantidad_puntas_sobrantes=Cantidad_menor;
            v_parejas=sum([Resultado_2.Numero_varillas_utilizadas]); % sumatoria de las varillas
utlizadas en esta sección
            vS_parejas=sum([Resultado_2.Cantidad_puntas_sobrantes]); % sumatoria de las
longitudes de puntas sobrantes en esta sección
            l=l+1;
```

```
        if Datos(j,3)==0
          Datos(j,2) =0;
        end
        if Datos (i,3)==0
           Datos(i,2)=0;
        end
        Datos;
      end
   end
end
l=l-1;
%% Trios
for b= 1:length(Datos)
   if Datos(b,3)~=0
     Indice_1A=Datos(b,2);
   else
      continue
   end
   for c=1:length(Datos)
      if b==c
         continue
      elseif Datos(c,3)~=0
          Indice_1B=Datos(c,2);
      else
         continue
      end
      Indice_1B=Datos(c,2);
      for d=1:length(Datos)
         if b==d
```

```
            continue
        elseif c==d
            continue
        elseif Datos (d,3)~=0
            Indice_1C=Datos(d,2);
        else
            continue
        end
        Diferencia=Insumo-(Indice_1A+Indice_1B+Indice_1C);
        if Diferencia <=Rango_P_S && Diferencia >=Rango_P_I && Datos(b,3)>0 &&
Datos(c,3)>0 && Datos(d,3)>0
            v=[Datos(b,3),Datos(c,3),Datos(d,3)];
            cantidad_menor2=min(v);
            Datos(b,3)=Datos(b,3)-cantidad_menor2;
            Datos(c,3)=Datos(c,3)-cantidad_menor2;
            Datos(d,3)=Datos(d,3)-cantidad_menor2;

            Resultado_3(m). Trabajo1= Datos(b,1);
            Resultado_3(m). Tipo_Corte1= Datos(b,2);
            Resultado_3(m). Cantidad1= v(1);
            Resultado_3(m). Sobrante1= Datos(b,3);

            Resultado_3(m). Trabajo2= Datos(c,1);
            Resultado_3(m). Tipo_Corte2= Datos(c,2);
            Resultado_3(m). Cantidad2= v(2);
            Resultado_3(m). Sobrante2= Datos(c,3);

            Resultado_3(m). Trabajo3= Datos(d,1);
            Resultado_3(m). Tipo_Corte3= Datos(d,2);
```

```
            Resultado_3(m). Cantidad3= v(3);
            Resultado_3(m). Sobrante3= Datos(d,3);

            Resultado_3(m). Numero_varillas_utilizadas= cantidad_menor2;
            Resultado_3(m). Longitud_punta=Diferencia;
            Resultado_3(m). Cantidad_puntas_sobrantes=cantidad_menor2;

            if Datos (b,3)==0
              Datos(b,2)=0;
            end
            if Datos(c,3)==0
              Datos(c,2)=0;
            end
            if Datos(d,3)==0
              Datos(d,2)=0;
            end
            m=m+1;
            v_trios=sum([Resultado_3.Numero_varillas_utilizadas]); % sumatoria de las varillas
utilizadas en esta sección
            vS_trios=sum([Resultado_3.Cantidad_puntas_sobrantes]); % sumatoria de las
longitudes de puntas sobrantes en esta sección
          continue
        end
      end
   end
end
m=m-1; % número de tríos formados
%% Cuartetos
for b_1= 1:length(Datos)
```

```
if Datos(b_1,3)~=0
  Indice_A=Datos(b_1,2);
else
   continue
end

for c_1=1:length(Datos)
   if b_1==c_1
      continue
   elseif Datos(c_1,3)~=0
       Indice_B=Datos(c_1,2);
   else
      continue
   end
   for d_1=1:length(Datos)
      if b_1==d_1
         continue
      elseif c_1==d_1
         continue
      elseif Datos (d_1,3)~=0
         Indice_C=Datos(d_1,2);
      else
         continue
      end
      for f_1=1:length(Datos);
         if b_1==f_1
            continue
         elseif c_1==f_1
            continue
```

```
        elseif d_1==f_1
            continue
        elseif Datos(f_1,3)~=0
            Indice_D=Datos(f_1,2);
        else
            continue
        end
        Diferencia=Insumo-(Indice_A+Indice_B+Indice_C+Indice_D);
        if  Diferencia <=Rango_P_S && Diferencia >=Rango_P_I  && Datos(b_1,3)>0 &&
Datos(c_1,3)>0 && Datos(d_1,3)>0 && Datos(f_1,3)>0

        v=[Datos(b_1,3),Datos(c_1,3),Datos(d_1,3),Datos(f_1,3)];
        cantidad_menor2=min(v);
        Datos(b_1,3)=Datos(b_1,3)-cantidad_menor2;
        Datos(c_1,3)=Datos(c_1,3)-cantidad_menor2;
        Datos(d_1,3)=Datos(d_1,3)-cantidad_menor2;
        Datos(f_1,3)=Datos(f_1,3)-cantidad_menor2;

        Resultado_4(p). Trabajo1= Datos(b_1,1);
        Resultado_4(p). Tipo_Corte1= Datos(b_1,2);
        Resultado_4(p). Cantidad1= v(1);
        Resultado_4(p). Sobrante1= Datos(b_1,3);

        Resultado_4(p). Trabajo2= Datos(c_1,1);
        Resultado_4(p). Tipo_Corte2= Datos(c_1,2);
        Resultado_4(p). Cantidad2= v(2);
        Resultado_4(p). Sobrante2= Datos(c_1,3);

        Resultado_4(p). Trabajo3= Datos(d_1,1);
```

```
            Resultado_4(p). Tipo_Corte3= Datos(d_1,2);
            Resultado_4(p). Cantidad3= v(3);
            Resultado_4(p). Sobrante3= Datos(d_1,3);

            Resultado_4(p). Trabajo4= Datos(f_1,1);
            Resultado_4(p). Tipo_Corte4= Datos(f_1,2);
            Resultado_4(p). Cantidad4= v(4);
            Resultado_4(p). Sobrante4= Datos(f_1,3);

            Resultado_4(p). Numero_varillas_utilizadas= cantidad_menor2;
            Resultado_4(p). Longitud_punta=Diferencia;
            Resultado_4(p). Cantidad_puntas_sobrantes=cantidad_menor2;

            if Datos (b_1,3)==0
              Datos(b_1,2)=0;
            elseif Datos(c_1,3)==0
               Datos(c_1,2)=0;
            elseif Datos(d_1,3)==0
               Datos(d_1,2)=0;
            elseif Datos(f_1,3)==0
               Datos(f_1,2)=0;
            end
            p=p+1;
            v_cuarteto=sum([Resultado_4.Numero_varillas_utilizadas]); % sumatoria de las
varillas utilizadas en esta sección
            vS_cuarteto=sum([Resultado_4.Cantidad_puntas_sobrantes]); % sumatoria de las
longitudes de puntas sobrantes en esta sección
           continue
            end
```

```
        end

    end

  end

end
p=p-1;% Numero de cuartetos armados
%% Quintetos
for b_2= 1:length(Datos)
  if Datos(b_2,3)~=0
    Indice_A=Datos(b_2,2);
  else
    continue
  end

  for c_2=1:length(Datos)
    if b_2==c_2
      continue
    elseif Datos(c_2,3)~=0
      Indice_B=Datos(c_2,2);
    else
      continue
    end
    for d_2=1:length(Datos)
      if b_2==d_2
        continue
      elseif c_2==d_2
        continue
```

```
elseif Datos (d_2,3)~=0
   Indice_C=Datos(d_2,2);
else
   continue
end
for f_2=1:length(Datos);
   if b_2==f_2
      continue
   elseif c_2==f_2
      continue
   elseif d_2==f_2
      continue
   elseif Datos(f_2,3)~=0
      Indice_D=Datos(f_2,2);
   else
      continue
   end

   for e_2=1:length(Datos)
      if e_2==b_2
         continue
      elseif c_2==e_2
         continue
      elseif d_2==e_2
         continue
      elseif f_2==e_2
         continue
      elseif Datos(e_2,3)~=0
         Indice_E=Datos(e_2,2);
```

```
            else
               continue
            end
            Diferencia=Insumo-(Indice_A+Indice_B+Indice_C+Indice_D+Indice_E);
            if Diferencia <=Rango_P_S && Diferencia >=Rango_P_I && Datos(b_2,3)>0 &&
Datos(c_2,3)>0 && Datos(d_2,3)>0 && Datos(f_2,3)>0 && Datos(e_2,3)>0

               v=[Datos(b_2,3),Datos(c_2,3),Datos(d_2,3),Datos(f_2,3),Datos(e_2,3)];
               cantidad_menor2=min(v);
               Datos(b_2,3)=Datos(b_2,3)-cantidad_menor2;
               Datos(c_2,3)=Datos(c_2,3)-cantidad_menor2;
               Datos(d_2,3)=Datos(d_2,3)-cantidad_menor2;
               Datos(f_2,3)=Datos(f_2,3)-cantidad_menor2;
               Datos(e_2,3)=Datos(e_2,3)-cantidad_menor2;

               Resultado_5(q). Trabajo1= Datos(b_2,1);
               Resultado_5(q). Tipo_Corte1= Datos(b_2,2);
               Resultado_5(q). Cantidad1= v(1);
               Resultado_5(q). Sobrante1= Datos(b_2,3);

               Resultado_5(q). Trabajo2= Datos(c_2,1);
               Resultado_5(q). Tipo_Corte2= Datos(c_2,2);
               Resultado_5(q). Cantidad2= v(2);
               Resultado_5(q). Sobrante2= Datos(c_2,3);

               Resultado_5(q). Trabajo3= Datos(d_2,1);
               Resultado_5(q). Tipo_Corte3= Datos(d_2,2);
               Resultado_5(q). Cantidad3= v(3);
               Resultado_5(q). Sobrante3= Datos(d_2,3);
```

```
Resultado_5(q). Trabajo4= Datos(f_2,1);
Resultado_5(q). Tipo_Corte4= Datos(f_2,2);
Resultado_5(q). Cantidad4= v(4);
Resultado_5(q). Sobrante4= Datos(f_2,3);

Resultado_5(q). Trabajo5= Datos(e_2,1);
Resultado_5(q). Tipo_Corte5= Datos(e_2,2);
Resultado_5(q). Cantidad5= v(5);
Resultado_5(q). Sobrante5= Datos(e_2,3);

Resultado_5(q). Numero_varillas_utilizadas= cantidad_menor2;
Resultado_5(q). Longitud_punta=Diferencia;
Resultado_5(q). Cantidad_puntas_sobrantes=cantidad_menor2;

if Datos (b_2,3)==0
   Datos(b_2,2)=0;
elseif Datos(c_2,3)==0
   Datos(c_2,2)=0;
elseif Datos(d_2,3)==0
   Datos(d_2,2)=0;
elseif Datos(f_2,3)==0
   Datos(f_2,2)=0;
elseif Datos (e_2,3)==0
   Datos(e_2,2)=0
end
q=q+1;
v_quinteto=sum([Resultado_5. Numero_varillas_utilizadas]); % suma el número de varillas utilizadas en esta sección
```

```
                    vS_quinteto=sum([Resultado_5.Cantidad_puntas_sobrantes]); % sumatoria de las
longitudes de puntas sobrantes en esta sección
                        continue
                    end
                end

            end

        end
    end
end
q=q-1;% Numero de cuartetos armados
%% sextetos
for b_3= 1:length(Datos)
    if Datos(b_3,3)~=0
        Indice_A=Datos(b_3,2);
    else
        continue
    end

    for c_3=1:length(Datos)
        if b_3==c_3
            continue
        elseif Datos(c_3,3)~=0
            Indice_B=Datos(c_3,2);
        else
            continue
        end
        for d_3=1:length(Datos)
```

```
if b_3==d_3
   continue
elseif c_3==d_3
   continue
elseif Datos (d_3,3)~=0
   Indice_C=Datos(d_3,2);
else
   continue
end
for f_3=1:length(Datos);
   if b_3==f_3
      continue
   elseif c_3==f_3
      continue
   elseif d_3==f_3
      continue
   elseif Datos(f_3,3)~=0
      Indice_D=Datos(f_3,2);
   else
      continue
   end

   for e_3=1:length(Datos)
      if e_3==b_3
         continue
      elseif c_3==e_3
         continue
      elseif d_3==e_3
         continue
```

```
elseif f_3==e_3
   continue
elseif Datos(e_3,3)~=0
   Indice_E=Datos(e_3,2);
else
   continue
end
for g_3=1:length(Datos)
   if g_3==b_3
      continue
   elseif g_3==c_3
      continue
   elseif g_3==d_3
      continue
   elseif g_3==f_3
      continue
   elseif g_3==e_3
      continue
   elseif Datos(g_3,3)~=0
      Indice_G=Datos(g_3,2);
   else
      continue
   end
   Diferencia=Insumo-(Indice_A+Indice_B+Indice_C+Indice_D+Indice_E+
Indice_G);
   if  Diferencia <=Rango_P_S && Diferencia >=Rango_P_I  && Datos(b_3,3)>0
&& Datos(c_3,3)>0 && Datos(d_3,3)>0 && Datos(f_3,3)>0 && Datos(e_3,3)>0&&
Datos(g_3,3)>0
```

```
v=[Datos(b_3,3),Datos(c_3,3),Datos(d_3,3),Datos(f_3,3),Datos(e_3,3),Datos(g_3,3)];
                cantidad_menor2=min(v);
                Datos(b_3,3)=Datos(b_3,3)-cantidad_menor2;
                Datos(c_3,3)=Datos(c_3,3)-cantidad_menor2;
                Datos(d_3,3)=Datos(d_3,3)-cantidad_menor2;
                Datos(f_3,3)=Datos(f_3,3)-cantidad_menor2;
                Datos(e_3,3)=Datos(e_3,3)-cantidad_menor2;
                Datos(g_3,3)=Datos(g_3,3)-cantidad_menor2;

                Resultado_6(s). Trabajo1= Datos(b_3,1);
                Resultado_6(s). Tipo_Corte1= Datos(b_3,2);
                Resultado_6(s). Cantidad1= v(1);
                Resultado_6(s). Sobrante1= Datos(b_3,3);

                Resultado_6(s). Trabajo2= Datos(c_3,1);
                Resultado_6(s). Tipo_Corte2= Datos(c_3,2);
                Resultado_6(s). Cantidad2= v(2);
                Resultado_6(s). Sobrante2= Datos(c_3,3);

                Resultado_6(s). Trabajo3= Datos(d_3,1);
                Resultado_6(s). Tipo_Corte3= Datos(d_3,2);
                Resultado_6(s). Cantidad3= v(3);
                Resultado_6(s). Sobrante3= Datos(d_3,3);

                Resultado_6(s). Trabajo4= Datos(f_3,1);
                Resultado_6(s). Tipo_Corte4= Datos(f_3,2);
                Resultado_6(s). Cantidad4= v(4);
                Resultado_6(s). Sobrante4= Datos(f_3,3);
```

```
Resultado_6(s). Trabajo5= Datos(e_3,1);
Resultado_6(s). Tipo_Corte5= Datos(e_3,2);
Resultado_6(s). Cantidad5= v(5);
Resultado_6(s). Sobrante5= Datos(e_3,3);

Resultado_6(s). Trabajo6= Datos(g_3,1);
Resultado_6(s). Tipo_Corte6= Datos(g_3,2);
Resultado_6(s). Cantidad6= v(6);
Resultado_6(s). Sobrante6= Datos(g_3,3);

Resultado_6(s). Numero_varillas_utilizadas= cantidad_menor2;
Resultado_6(s). Longitud_punta=Diferencia;
Resultado_6(s). Cantidad_puntas_sobrantes=cantidad_menor2;

if Datos (b_3,3)==0
   Datos(b_3,2)=0;
elseif Datos(c_3,3)==0
   Datos(c_3,2)=0;
elseif Datos(d_3,3)==0
   Datos(d_3,2)=0;
elseif Datos(f_3,3)==0
   Datos(f_3,2)=0;
elseif Datos (e_3,3)==0
   Datos(e_3,2)=0;
elseif Datos(g_3,3)==0
   Datos(g_3,2)=0;
end
s=s+1;
```

```
                    v_sexteto=sum([Resultado_6. Numero_varillas_utilizadas]); % Suma el
número de varillas de esta sección.
                    vS_sexteto=sum([Resultado_6.Cantidad_puntas_sobrantes]); % sumatoria de
las longitudes de puntas sobrantes en esta sección
                    continue
                end
            end


        end


    end
  end
end
s=s-1;% Numero de cuartetos armados
%% Cantidad
Fijador=0;
for kk=1:length(Datos)
  if Datos(kk,2)~=0 && Datos(kk,3)~=0
  for DD=1:length(Datos) %Numero mínimo de veces para poder efectuar los cortes bajo el
rango establecido.(por varilla)
    Diferenciador=Datos(kk,2)*DD;
    if  Diferenciador <= (Insumo-Rango_P_I) && Diferenciador >=(Insumo-Rango_P_S)
      Fijador=DD; % indica el número de unificaciones posibles que se pueden hacer para
poder estar dentro del rango aceptable
      break
    else
      continue
    end
```

```
end
Datos(kk,2);
Fijador;
Residuo=Datos(kk,3);
if Residuo < Fijador
   break
end
for DDD=1:Datos(kk,3)
   Residuo=Residuo-Fijador;
   if Residuo < Fijador
      break
   end
end
if Fijador==0 % Si la cantidad del corte no es suficiente para unirlo y entrar en el rango.
   DDD=0;
   Punta_sobrante=0;
else
  Punta_sobrante=Insumo-(Fijador*Datos(kk,2));
end
Datos(kk,2);
Datos(kk,3);
DDD;
Fijador;
Residuo;
Resultado_Cantidad(mm). Trabajo= Datos(kk,1);
Resultado_Cantidad(mm). Tipo__Corte=Datos(kk,2);
Resultado_Cantidad(mm). Cantidad=Datos(kk,3);
Resultado_Cantidad(mm). Sobrante=Residuo;
Resultado_Cantidad(mm). Cantidad_cortes= Fijador*DDD;     % indica el número de cortes
```

```
realizados
   Resultado_Cantidad(mm). Numero_varillas_utilizadas=DDD;



   Resultado_Cantidad(mm). Longitud_punta=Punta_sobrante;
   Resultado_Cantidad(mm). Cantidad_puntas_sobrantes=DDD;

   vS_cantidad=sum([Resultado_Cantidad.Cantidad_puntas_sobrantes]); % sumatoria de las
longitudes de puntas sobrantes en esta sección
   v_cantidad=sum([Resultado_Cantidad. Numero_varillas_utilizadas]); % numero de varillas
utilizadas en esta sección.
   Sum_cortes_realizados_cantidad=sum([Resultado_Cantidad. Cantidad_cortes]); % Número
total de cortes realizados en esta sección.
   mm=mm+1;
   Datos(kk,3)=Residuo;
   end
   if Datos(kk,3)==0
      Datos (kk,2)=0;
   end
end
mm=mm-1; % Numero de Cantidades formadas
%% Resultado
% Números de trabajos terminados.
Datos;
m_comparacion=zeros(length(Datos),3);
ff=1;
fprintf('\nTrabajos terminados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad \n');
for jj=1:length(Datos)
```

```
    if Datos(jj,2)==0
        m_comparacion(ff,1)=Datos_2(jj,1);
        m_comparacion(ff,2)=Datos_2(jj,2);
        m_comparacion(ff,3)=Datos_2(jj,3);
        fprintf('%6d %12.2f %11d \n', m_comparacion(ff,1),
m_comparacion(ff,2),m_comparacion(ff,3));
        ff=1+ff;
    end
end
fprintf('\nNumero de trabajos terminados:\t');
fprintf('%1d\n',ff-1);

% Numero de trabajos empezados= Comparado con la matriz de datos inicial
% del problema.
Datos;
Datos_2;
n_comparador=zeros(length(Datos_2),4);
gg=0;
fprintf('\nTrabajos empezados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad Realizada  Cantidad Faltante  \n');
for ii=1:length(Datos)
    if Datos(ii,2)~=0
       gg;
        if Datos (ii,3)~= Datos_2(ii,3)
           gg=gg+1;
           n_comparador(gg,1)= Datos(ii,1); % N° del trabajo respectivo
           n_comparador(gg,2)= Datos(ii,2); % Tipo de corte
           n_comparador(gg,3)= Datos_2(ii,3)-Datos(ii,3); % Cantidad Realizada
           n_comparador(gg,4)= Datos(ii,3); % Cantidad Faltante
```

```
        fprintf('%6d %12.2f %16d %19d \n',n_comparador(gg,1),
n_comparador(gg,2),n_comparador(gg,3),n_comparador(gg,4));
      end
   end
end
n_comparador; %Matriz almacenadora
disp('Numero de trabajos empezados: ');
disp(gg);
disp('Sumatoria de las cantidades termiandas en los trabajos emprezados:');
disp(sum(n_comparador(:,3)));
disp('Sumatoria de las cantidades faltantes en los trabajos empezados:');
disp(sum(n_comparador(:,4)));

% Numero de trabajos NO tocados = Comparado con la matriz de datos inicial
% del problema.
Datos;
Datos_2;
y_comparador=zeros(length(Datos_2),3);
cc=0;
fprintf('\nTrabajos No tocados:\n');
fprintf('\n Posicion  Tipo de Corte  Cantidad \n');
for yy=1:length(Datos)
   if Datos(yy,2)~=0
      cc;
       if Datos (yy,3)== Datos_2(yy,3)
          cc=cc+1;
          y_comparador(cc,1)= Datos(yy,1); % N° del trabajo respectivo
          y_comparador(cc,2)= Datos(yy,2); % Tipo de corte
          y_comparador(cc,3)= Datos(yy,3); % Cantidad sin tocar
```

```
        fprintf('%6d %12.2f %11d \n', y_comparador(cc,1),
y_comparador(cc,2),y_comparador(cc,3));
      end
   end
end
y_comparador;
disp('Numero de trabajos no tocados:')
disp(cc);
disp('Sumatoria de las cantidad de cortes no tocados:');
disp(sum(y_comparador(:,3)));

% Número total de varillas utilizadas.
Total_Varillas=sum([v_unidad,v_parejas,v_trios,v_cuarteto, v_quinteto, v_sexteto, v_cantidad]);
% Cantidad total de varillas utilizadas en este Scrip
fprintf('\nNumero Total de Varillas utilizadas es:\t');
fprintf('%1d\n',Total_Varillas);

% Número total de longitudes sobrantes.
sobrantes_totales=sum([vS_unidad,vS_cantidad,vS_parejas,vS_trios,vS_cuarteto,vS_quinteto,vS
_sexteto]); % Cantidad total de sobrantes
fprintf('\nNumero Total de de longitudes sobrantes es:\t');
fprintf('%d\n',sobrantes_totales);
 % Numero de cortes totales realizados.
Numero_cortes_totales=Sum_cortes_realizados_cantidad+vS_unidad*1+vS_parejas*2+vS_trios
*3+vS_cuarteto*4+vS_quinteto*5+vS_sexteto*6;
fprintf('\nNumero Total de Cortes realizados:\t');
fprintf('%1d\n',Numero_cortes_totales);
```

Código fuente en Matlab de la Heurística de Johnson

```
tic
clear all
clc
Datos= xlsread('Datos','Hoja1','A1:C20000');
Secuencia_resultado=zeros(1,length(Datos));
% Datos_procesar=Datos(1:10,2:3);
j=length(Datos); % Guía para asignar trabajos de manera descendente
k=1; % Guía para asignar trabajos de manera ascendente
for i=1:length(Datos);
   Datos; %llamado de la matriz.
   [V,I]=min(Datos,[],1); %buscar el valor y el índice del menor escalar en la matriz llamada
   if k==j % si las posiciones en ambos extremos llegan a encontrase asignar de manera arbitraria el trabajo en esa posición
      Secuencia_resultado(j)=Datos(I(2));
      break
   end
   if V(3)<=V(2) % si el tp de la maquina 1 es menor que el de la maquina 2 para ese trabajo se le asigna en la segunda parte
      for p=1:length(Datos)
         j;
         Secuencia_resultado(j)=Datos(I(3));
         Datos(I(3),:)=[];
         j=j-1;
         break
      end
   else  % si el tp de la maquina 2 es menor que el de la maquina 1 para ese trabajo se le
```

```
asigna en la primera  parte
      for w=1:length(Datos)
         k;
         Secuencia_resultado(k)=Datos(I(2));
         Datos(I(2),:)=[];
         k=1+k;
         break
      end
   end
end
Secuencia_resultado;

%% Evaluación del Makespanpara dicha secuencia.
Secuencia_resultado;
Datos= xlsread('Datos','Hoja1','A1:C20000');
s=Secuencia_resultado';
[Fila,Columna]=size(Datos);
Tiempo_procesado=zeros(Fila,2); % separador de memoria para asignar los tiempos de procesados para el respecto trabajo.

for q=1:length(s) % asignación de los tiempos de procesado para los trabajos organizados según la heurística
   Indice=s(q);
   Tiempo_procesado(q,:)=Datos(Indice,2:3);
end
s=[s,Tiempo_procesado];
Tiempo_terminacion_1= cumsum(s(:,2)); % Tiempos de terminación de los trabajos de la maquina 1
Tiempo_terminacion_2=zeros(2,length(s))'; % almacenador de los tiempos de
```

```
terminación de los trabajos en la maquina 2

for w=1:length(s) % Asignación de los tiempos de inicio y fin de los trabajos de la
maquina 2
   if w==1 % tiene un proceso diferentes porque es el primer trabajo en asignar
      Tiempo_terminacion_2(1,1)= Tiempo_terminacion_1(1);
      Tiempo_terminacion_2(1,2)= Tiempo_terminacion_1(1)+Tiempo_procesado(1,2);
   else
       if Tiempo_terminacion_2(w-1,2)>Tiempo_terminacion_1(w)
          Tiempo_terminacion_2(w,1)= Tiempo_terminacion_2(w-1,2)+1;
          Tiempo_terminacion_2(w,2)=
Tiempo_terminacion_2(w,1)+Tiempo_procesado(w,2)-1;
       else
          Tiempo_terminacion_2(w,1)= Tiempo_terminacion_1(w)+1;
          Tiempo_terminacion_2(w,2)=
Tiempo_terminacion_2(w,1)+Tiempo_procesado(w,2)-1;
       end
   end
end

Tiempo_terminacion_2;
Tiempos_terminacion_total=[Tiempo_terminacion_1,Tiempo_terminacion_2(:,2)];
Makespan=max(Tiempos_terminacion_total(Fila,:));
%% Resultado...
Secuencia_resultado
Makespan
toc
```

Código fuente en Matlab de la Regla de Despacho EDD

```
function [Orden_EDD,Vector_Resultado_EDD]=EDD(Datos)
%% EDD (earliest due date first, Primero el plazo mas próximo)
clear all
clc
Datos=xlsread ('Datos','Hoja','B2:D20000');
[f,c]=size(Datos);
[Orden,Indice] = sort(Datos(:,3));
Orden = Datos(Indice,:);
Vector_Resultado_EDD=zeros(1,9)';

%% Evaluador
Orden;
v_r=zeros(1,length(Orden))';
for a=1:length(Orden)
   if a==1
     v_r(a)=Orden(a,2);
   else
      v_r(a)=v_r(a-1)+Orden(a,2);
   end
end
Orden=[Orden,v_r];
v_r2=zeros(1,length(Orden))';
for b=1:length(Orden)
   v_r2(b)= Orden(b,3)-Orden(b,4);
end
Orden=[Orden,v_r2];
```

```
%% Numero de trabajos.
f;
Vector_Resultado_EDD(1)=f;
%% Sumatoria de los tiempos de proceso(Makespan)
Sum_tp=sum(Orden(:,2));
Vector_Resultado_EDD(2)=Sum_tp;

%% Contador de trabajos tardíos y adelantados v_r2
p=0; % Contador de trabajos adelantados.
n=0; % Contador de trabajos tardíos.
sum_adelanto=0;
sum_atrazo=0;
for c=1:length(Orden)
   if Orden(c,5)<0
      n;
      n=n+1;
      sum_atrazo=Orden(c,5)+sum_atrazo;
   else
      p;
      p=p+1;
      sum_adelanto=Orden(c,5)+sum_adelanto;
   end
end
Vector_Resultado_EDD(3)=n; % Numero de trabajos tardíos
Vector_Resultado_EDD(4)=p; % Numero de trabajos adelantados

%% Sumatorio del tiempo flujo.
Sum_Tflujo=sum(Orden(:,4));
Vector_Resultado_EDD(9)=Sum_Tflujo;
```

```
%% Adelanto máximo= Max -- Tardanza máxima= Min
max_min=Orden(:,5);
Min=min(max_min);
Max=max(max_min);
Vector_Resultado_EDD(5)=abs(Min); % Tardanza Máxima
Vector_Resultado_EDD(6)=abs(Max); % Adelanto Máximo

%% Adelanto promedio y tardanza promedio.
A_pro=abs(sum_adelanto/f);
T_pro=abs(sum_atrazo/f);
Vector_Resultado_EDD(7)=T_pro;
Vector_Resultado_EDD(8)=A_pro;

%% resultado...
Orden_EDD=Orden
Vector_Resultado_EDD

end
```

Código fuente en Matlab de la regla de despacho FCFS

```
function [Orden_FCFS,Vector_Resultado_FCFS]=FCFS(Datos)
%% FCFS ( FIRST COME, FIRST SERVED)
```

```
clear all
clc
Datos=xlsread ('Datos','Hoja','B2:D20000');
[f,c]=size(Datos);
Orden = zeros(f,c);
Orden=Datos;
Vector_Resultado_FCFS=zeros(1,9)';
%% Evaluador
Orden;
v_r=zeros(1,length(Orden))';
for a=1:length(Orden)
   if a==1
     v_r(a)=Orden(a,2);
   else
      v_r(a)=v_r(a-1)+Orden(a,2);
   end
end
Orden=[Orden,v_r];
v_r2=zeros(1,length(Orden))';
for b=1:length(Orden)
   v_r2(b)= Orden(b,3)-Orden(b,4);
end
Orden=[Orden,v_r2];

%% Numero de trabajos.
f;
Vector_Resultado_FCFS(1)=f;

%% Sumatoria de los tiempos de proceso(Makespan)
```

```
Sum_tp=sum(Orden(:,2));
Vector_Resultado_FCFS(2)=Sum_tp;

%% Contador de trabajos tardíos y adelantados v_r2
p=0; % Contador de trabajos adelantados.
n=0; % Contador de trabajos tardíos.
sum_adelanto=0;
sum_atrazo=0;
for c=1:length(Orden)
   if Orden(c,5)<0
      n;
      n=n+1;
      sum_atrazo=Orden(c,5)+sum_atrazo;
   else
      p;
      p=p+1;
      sum_adelanto=Orden(c,5)+sum_adelanto;
   end
end
Vector_Resultado_FCFS(3)=n; % Numero de trabajos tardíos
Vector_Resultado_FCFS(4)=p; % Numero de trabajos adelantados

%% Sumatorio del tiempo flujo.
Sum_Tflujo=sum(Orden(:,4));
Vector_Resultado_FCFS(9)=Sum_Tflujo;

%% Adelanto máximo= Max -- Tardanza máxima= Min
max_min=Orden(:,5);
Min=min(max_min);
```

```
Max=max(max_min);
Vector_Resultado_FCFS(5)=abs(Min); % Tardanza Máxima
Vector_Resultado_FCFS(6)=abs(Max); % Adelanto Máximo

%% Adelanto promedio y tardanza promedio.
A_pro=abs(sum_adelanto/f);
T_pro=abs(sum_atrazo/f);
Vector_Resultado_FCFS(7)=T_pro;
Vector_Resultado_FCFS(8)=A_pro;

%% resultado...
Orden_FCFS=Orden;
Vector_Resultado_FCFS;

end
```

Código fuente en Matlab de la regla de despacho SOT

```
function [Orden_SOT,Vector_Resultado_SOT]=SOT(Datos)
%% SOT(shortest operating time, tiempo de operación más breve)
```

```
clear all
clc
Datos=xlsread ('Datos','Hoja','B2:D20000');
[f,c]=size(Datos);
[Orden,Indice] = sort(Datos(:,2));
Orden = Datos(Indice,:);
Vector_Resultado_SOT=zeros(1,9)';
%% Evaluador
Orden;
v_r=zeros(1,length(Orden))';
for a=1:length(Orden)
   if a==1
     v_r(a)=Orden(a,2);
   else
      v_r(a)=v_r(a-1)+Orden(a,2);
   end
end
Orden=[Orden,v_r];
v_r2=zeros(1,length(Orden))';
for b=1:length(Orden)
   v_r2(b)= Orden(b,3)-Orden(b,4);
end
Orden=[Orden,v_r2];
%% Numero de trabajos.
f;
Vector_Resultado_SOT(1)=f;
%% Sumatoria de los tiempos de proceso(Makespan)
Sum_tp=sum(Orden(:,2));
Vector_Resultado_SOT(2)=Sum_tp;
```

```
%% Contador de trabajos tardíos y adelantados v_r2
p=0; % Contador de trabajos adelantados.
n=0; % Contador de trabajos tardíos.
sum_adelanto=0;
sum_atrazo=0;
for c=1:length(Orden)
   if Orden(c,5)<0
      n;
      n=n+1;
      sum_atrazo=Orden(c,5)+sum_atrazo;
   else
      p;
      p=p+1;
      sum_adelanto=Orden(c,5)+sum_adelanto;
   end
end
Vector_Resultado_SOT(3)=n; % Numero de trabajos tardíos
Vector_Resultado_SOT(4)=p; % Numero de trabajos adelantados
%% Sumatorio del tiempo flujo.
Sum_Tflujo=sum(Orden(:,4));
Vector_Resultado_SOT(9)=Sum_Tflujo;
%% Adelanto máximo= Max -- Tardanza máxima= Min
max_min=Orden(:,5);
Min=min(max_min);
Max=max(max_min);
Vector_Resultado_SOT(5)=abs(Min); % Tardanza Máxima
Vector_Resultado_SOT(6)=abs(Max); % Adelanto Máximo
%% Adelanto promedio y tardanza promedio.
A_pro=abs(sum_adelanto/f);
```

```
T_pro=abs(sum_atrazo/f);
Vector_Resultado_SOT(7)=T_pro;
Vector_Resultado_SOT(8)=A_pro;
%% resultado...
Orden_SOT=Orden
Vector_Resultado_SOT
end
```

Código fuente en Matlab de la regla de despacho STR

```
function [Orden_STR,Vector_Resultado_STR]=STR(Datos)
%% STR
clear all
clc
Datos=xlsread ('Datos','Hoja','B2:D20000');
[f,c]=size(Datos);
Diferenciador=zeros(1,f);
for qqq=1:length(Datos)

   Diferenciador(qqq)=Datos(qqq,3)-Datos(qqq,2);
end
Datos=[Datos,Diferenciador'];
[Orden,Indice] = sort(Datos(:,4));
Orden = Datos(Indice,:);
Orden(:,4)=[];
Vector_Resultado_STR=zeros(1,9)';
%% Evaluador
Orden;
```

```
v_r=zeros(1,length(Orden))';
for a=1:length(Orden)
   if a==1
     v_r(a)=Orden(a,2);
   else
      v_r(a)=v_r(a-1)+Orden(a,2);
   end
end
Orden=[Orden,v_r];
v_r2=zeros(1,length(Orden))';
for b=1:length(Orden)
   v_r2(b)= Orden(b,3)-Orden(b,4);
end
Orden=[Orden,v_r2];
%% Numero de trabajos.
f;
Vector_Resultado_STR(1)=f;
%% Sumatoria de los tiempos de proceso(Makespan)
Sum_tp=sum(Orden(:,2));
Vector_Resultado_STR(2)=Sum_tp;
%% Contador de trabajos tardíos y adelantados v_r2
p=0; % Contador de trabajos adelantados.
n=0; % Contador de trabajos tardíos.
sum_adelanto=0;
sum_atrazo=0;
for c=1:f
   if Orden(c,5)<0
      n;
      n=n+1;
```

```
        sum_atrazo=Orden(c,5)+sum_atrazo;
    else
        p;
        p=p+1;
        sum_adelanto=Orden(c,5)+sum_adelanto;
    end
end
Vector_Resultado_STR(3)=n; % Numero de trabajos tardíos
Vector_Resultado_STR(4)=p; % Numero de trabajos adelantados
%% Sumatorio del tiempo flujo.
Sum_Tflujo=sum(Orden(:,4));
Vector_Resultado_STR(9)=Sum_Tflujo;
%% Adelanto maximo= Max -- Tardanza máxima= Min
max_min=Orden(:,5);
Min=min(max_min);
Max=max(max_min);
Vector_Resultado_STR(5)=abs(Min); % Tardanza Máxima
Vector_Resultado_STR(6)=abs(Max); % Adelanto Máximo
%% Adelanto promedio y tardanza promedio.
A_pro=abs(sum_adelanto/f);
T_pro=abs(sum_atrazo/f);
Vector_Resultado_STR(7)=T_pro;
Vector_Resultado_STR(8)=A_pro;
%% resultado...
Orden_STR=Orden;
Vector_Resultado_STR;
end
```

Código fuente en Matlab de la evaluación de las reglas de despacho

```
%% Reglas de despacho
clear all
clc
Datos=xlsread ('Datos','Hoja','B2:D20000');
[Orden_FCFS,Vector_Resultado_FCFS]=FCFS(Datos);
[Orden_SOT,Vector_Resultado_SOT]=SOT(Datos);
[Orden_EDD,Vector_Resultado_EDD]=EDD(Datos);
[Orden_STR,Vector_Resultado_STR]=STR(Datos);
R_D=zeros(9,4);
R_D=[Vector_Resultado_FCFS,Vector_Resultado_SOT,Vector_Resultado_EDD,Vector_Result
ado_STR];
% fprintf('\n Reglas de despacho\n');
fprintf('\vNumero de trabajos\v  Tiempo de terminacion\v  Numero de trabajos tardios\v
Numero de trabajos adelantados\v  Tardanza maxima\v  Adelanto maximo\v  Tardanza
promedio\v  Adelanto promedio\v  Sumatoria del tiempo flujo\v');
fprintf('\n %9.2f %23.2f %23.2f %31.2f %27.2f %16.2f %17.2f %20.2f
%24.2f\n\n',Vector_Resultado_FCFS,Vector_Resultado_SOT,Vector_Resultado_EDD,Vector_R
esultado_STR);
```