FORMULACIÓN DE HEURÍSTICA PARA MINIMIZAR EL *MAKESPAN* EN EL PROBLEMA DE *SCHEDULING JOB-SHOP* FLEXIBLE

Iván Esteban Valdés Guayara

Tutor:

Profesor. Andrés Alberto García León

Universidad De Ibagué

Programa de Ingeniería Industrial

2018

Tabla de contenido

RESUMEN 4
CAPITULO 1. FORMULACION DEL PROBLEMA 6
1.1 Definición del problema de Job-Shop Flexible 6
1.2 Complejidad del FJSP 7
1.3 Modelo conceptual para el FJSP 8
1.4 Representación por medio de grafos para el FJSP 9
1.5 Estado del Arte 9
1.6 Modelo de programación lineal para el FJSP 10
CAPÍTULO 2 FORMULACIÓN DE LA HEURÍSTICA 13
2.1 Solución inicial para el FJSP 13
2.2 Estructuras de vecindarios 14
2.3 Factibilidad del movimiento de operaciones críticas 15
2.4 Funciones de estimación 16
2.5 Funcionamiento de la heurística 16
CAPÍTULO 3. EXPERIMENTOS COMPUTACIONALES 17
3.1 Solución óptima al FJSP al minimizar *makespan* 17
3.2 Ejemplo de solución inicial 18
3.3 Calibración de parámetros a y b durante el proceso de diversificación 20
3.4 Cálculo del *makespan* para el FJSP 22
3.5 Comparación de la heurística con la programación lineal 23
CONCLUSIONES 25
REFERENCIAS BIBLIOGRÁFICAS 27

Índice de figuras:

Figura 1. Representación del FJSP por medio de un grafo 9
Figura 2. Ejemplo de vecindarios para el FJSP 15
Figura 3. Inserción de x entre j y k 15
Figura 4. Asignación de cargas con operaciones de una sola maquina 18
Figura 5. Asignación de máquinas a cada operación 19
Figura 6. Asignación y secuenciación completa 19
Figura 7. Grafo de la solución inicial del modelo de JSF 20

Índice de tablas:

Tabla 1. Solución óptima para instancias de 5 x 10 de (Hurink et al., 1994) 18
Tabla 2. Calibración de parámetros [a,b] en instancias de (Dauzere-Peres & Pauli., 1998) 21
Tabla 3. Cálculo del makespan para las instancias de (Dauzere-Peres & Pauli., 1998) 22
Tabla 4. Comparativo de la solución óptima y la heurística en la instancias de (Hurink et al., 1994) 23

RESUMEN

A lo largo de la historia los procesos productivos se han adaptado a las necesidades que imponen las condiciones socio-culturales; hace algunos años la necesidad por satisfacer grandes demandas conllevo a los procesos productivos a industrializarse y enfocarse en la producción masiva de un bien o servicio en específico. Sin embargo, hoy en día con la globalización de los mercados, la competencia y la necesidad por satisfacer los requerimientos específicos de cada cliente han obligado a las empresas desarrollar sistemas de producción que permitan generar especificaciones en los productos bajo pedidos.

El entorno moderno obliga a las organizaciones a desarrollar estructuras operativas flexibles y a que a su vez redunden en reducciones de costos, los cuales se manifiestan en el aprovechamiento de la maquinaria, la mano de obra y los recursos. En esta asistencia de investigación, se presenta a un problema que modela situaciones reales, el cual es el problema de *scheduling Job-shop* flexible, en el cual dos sub-problemas se deben resolver: Asignación y secuenciación. El primero determina para una operación la máquina que se le asignará; mientras para el segundo el orden de operaciones sobre las máquinas asignadas.

El objetivo de esta asistencia de investigación es diseñar una herramienta computacional para optimizar el makespan en el problema de *scheduling Job-shop* Flexible. Esta herramienta se basa principalmente en los resultados de (García-León et al, 2016). La construcción de esta heurística es fundamentada en un proceso de búsqueda local y para ello se propusieron tres objetivos específicos.

El primer objetivo específico consiste en formular funciones de estimación al mover operaciones críticas. El segundo en aplicar el modelo de PL al problema objeto de estudio en instancias pequeñas y finalmente formular el proceso de búsqueda local.

Para esta asistencia, no se ha considerado solucionar un problema específico vinculado a algún proceso industrial. Para los experimentos, se han considerado las instancias más complejas de solucionar que se encuentran en la literatura científica (Dauzère-Pérès & Paulli., 1998).

El documento está dividido en tres capítulos. En el primer Capítulo, se aborda el problema de *scehduling Job-Shop* flexible con la definición del criterio a optimizar (*makespan*). En el segundo Capítulo, se aborda la formulación matemática para optimizar criterios en el problema, en el tercer capítulo, se define el proceso de búsqueda local y finalmente los experimentos computacionales que validan los capítulos anteriores.

CAPITULO 1. FORMULACION DEL PROBLEMA

En este capítulo se detallan los aspectos sobresalientes del problema.

1.1 Definición del problema de Job-Shop Flexible

El problema de *scheduling Job-Shop Flexible* (FJSP) es considerado como uno de los problemas más complejos respecto a los problemas de programación de la producción. Dicha complejidad parte principalmente de su Flexibilidad para ejecutar cada una de las operaciones en diferentes maquinas posibles, que crean un conjunto de combinaciones bastante dilatadas.

La comprensión del FJSP, implica inicialmente definir el problema clásico de *scheduling Job-Shop*, en donde se tiene un conjunto de n trabajos J y un conjunto de m máquinas M. Cada trabajo consiste en una secuencia de operaciones, las cuales se realizan en una de las máquinas durante un tiempo determinado (tiempo de procesamiento). Una vez iniciada la operación en una máquina, se asume que esta no se puede interrumpir, de igual manera se determina que cada máquina realiza solamente una operación a la vez. La programación de órdenes entonces consiste en secuenciar las operaciones a las máquinas en un orden definido, con el fin de encontrar la secuencia que optimice un criterio y que para el caso particular corresponde al makespan, el cual representa el mayor tiempo de finalización de todos los pedidos.

Una extensión del Job-Shop es el Job-Shop Flexible, la cual considera que para cada una de las operaciones se tiene un subconjunto de máquinas para seleccionar una única y poder ser procesada. Por consiguiente, el FJSP es mucho más complejo y aquí se debe solucionar dos sub problemas: Asignación y Secuenciación. En la asignación se determina a que máquina se asignará una operación, mientras que la segunda determinará el orden en que se secuenciaran las operaciones en las máquinas asignadas para minimizar el *makespan*.

1.2 Complejidad del FJSP

La gran capacidad de combinaciones que se generan en el FJSP, debido a, sus características de flexibilidad al permitir realizar secuencias de operaciones en diversas máquinas, lo convierten en términos computacionales, y más específicamente en términos de tiempo y cálculos matemáticos, en un problemas demasiado complejo de solucionar, además dicha complejidad aumenta a medida que el modelo es más grande (en términos de trabajos y maquinas).

FJSP tiene características *NP-hard*, esto quiere decir que, inclusive cuando se presenta el caso particular de dos máquinas y tres pedidos, y un criterio de minimización del *makespan*. (Brucker, 2007). Existen muy pocos casos en los cuales el FJSP se puede solucionar polinomialmente. En términos computacionales, resolver un problema en tiempo polinomial, indica que el tiempo de ejecución de un algoritmo es menor a un cierto valor calculado dependiendo de las variables implicadas, y usando formulas polinómicas se determina que el problema se puede resolver en tiempo polinomial. Sin embargo, este no es el caso de la mayor parte de las extensiones de Job-Shop dado que al alterar sus variables puede conllevar a un aumento exponencial en sus requerimientos computacionales. Por tal razón [Brucker & Schlie, 1990] propusieron un algoritmo polinomial denominado "enfoque geométrico", en donde se programan dos trabajos que poseen tiempos de procesamiento que no dependen de la maquina donde se seleccionó, es decir, JSF con máquinas relacionadas.

Este enfoque se complementó luego por (Jurisch, 1995) al trabajar las fechas en las cuales se deben realizar las operaciones y las restricciones en las precedencias que hay entre las operaciones. [Mati & Xie, 2004] exponen que el FJSP con dos trabajos también es *NP-hard* cuando los tiempos de procesamiento de las operaciones dependen exclusivamente de la maquina asignada, en otras palabras, *Job-Shop* Flexible con máquinas no relacionadas. Además de eso hacen énfasis en el enfoque geométrico para minimizar criterios regulares del JSF con dos trabajos en donde solo uno es flexible.

1.3 Modelo conceptual para el FJSP

El FJS presenta un conjunto de máquinas ($i = 1, 2 \ldots n$), un conjunto de trabajos ($J = 1, 2 \ldots j$) y un conjunto de operaciones ($O = 1, 2 \ldots k$), en donde cada trabajo está conformado por un conjunto de operaciones (O € J), y dada la asignación cada operación posee una maquina asignada ($O_{j,k,n}$). Como parámetros existe un tiempo de procesamiento en cada operación que depende de la maquina asignada ($TP_{j,k,n}$).

El FJSP se opera baja las siguientes condiciones:

- ✓ Todos los trabajos están disponibles para empezar a ser procesados en el instante t = 0.
- ✓ Todas las máquinas se encuentran listas para la operación en $t = 0$.
- ✓ No se permite la interrupción de los trabajos en las máquinas, es decir, que una vez inicia una operación en una máquina ésta solamente queda disponible hasta que la operación finaliza.
- ✓ Hasta tanto una operación haya terminado su procesamiento, la máquina en la cual se esté realizando dicha operación no se podrá considerar disponible para ningún otro trabajo.
- ✓ Todos los trabajos tienen la misma prioridad dentro del sistema.
- ✓ Una vez finalice el proceso de una operación en cualquier máquina, ésta automáticamente queda disponible para recibir la siguiente operación. No se consideran tiempos de alistamiento.
- ✓ Las máquinas pueden realizar algunas operaciones.
- ✓ Los tiempos de procesamiento de las operaciones en las máquinas son enteros y conocidos.
- ✓ El tiempo total de proceso para un trabajo J es la suma del tiempo de proceso de cada una de las operaciones que lo componen. No se consideran alistamientos ni tiempos de parada.
- ✓ Existe restricción de secuencia en las operaciones de cada uno de los trabajos J, es decir, que la operación $O_{j,2,n}$ solamente podrá comenzar a ser procesada en la máquina respectiva una vez finalice completamente la operación $O_{j,1,n}$.

1.4 Representación por medio de grafos para el FJSP

FJSP puede ser representado mediante la utilizacion de un grafo, esta heramienta permite ilustrar el conjunto de pedidos, maquinas y operaciones. En la Figura 1, se muestra un ejemplo de un problema con tres pedidos y tres máquinas denominadas M_1, M_2 y M_3. El primero pedido tiene tres operaciones consecutivas (O_{11}, O_{12} y O_{13}), el segundo con dos y el tercero con tres. Cada operación debe ser realizada sobre una única máquina y debe seleccionar según las alternativas que posee. Por ejemplo para O_{11} se tiene que escoger una máquina para ser asisgnada y se debe seleccionara entre M_1 donde el tiempo de procesamiento es de 2 o M_2 con un tiempo de procesamiento de 3.

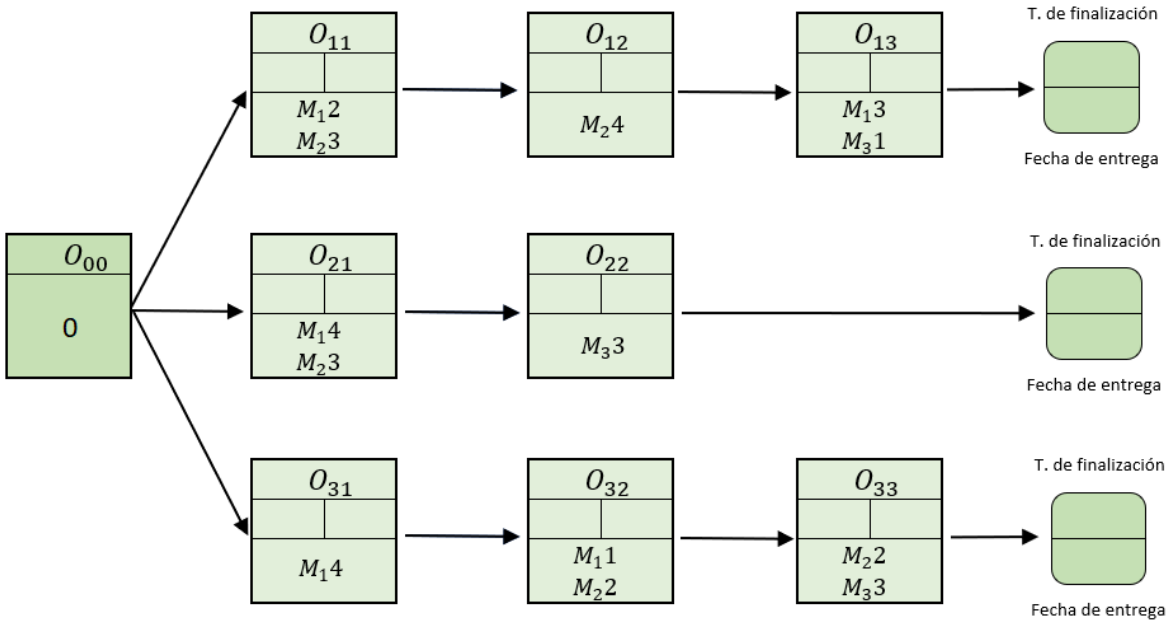

Figura 1. Representación del FJSP por medio de un grafo

Una vez realizada la asignación según reglas decisorias se procede a secuenciar. Ejemplo de ello se analizará en el algoritmo de solución inicial.

1.5 Estado del Arte

Numerosas investigaciones se han realizado entorno al problema FJSP. A lo largo del tiempo numerosos métodos y algoritmos han sido propuestos para dar solución a diferentes instancias del problema dependiendo de los criterios de dichas instancias.

Muchos de esos estudios se han enfocado en metodologías conocidas como algoritmos iterativos de búsqueda o técnicas conocidas como meta-heurísticas (Correa Rodríguez & Londoño, 2012).

En este apartado se toma un estado del arte reciente en torno al FJSP, el cual menciona que los artículos de los últimos años (1990-2014) han maximizado los esfuerzos por solucionar el problema (Chaundhry & Khan, 2015). Los autores encontraron 404 fuentes que incluyen 191 artículos, 167 documentos de conferencias, 12 tesis y 4 reportes. Basado en estas investigaciones se concluyó que: El número de publicaciones ha aumentado considerablemente en los últimos cinco años. Po otra parte, se encontró que el *makespan* es el criterio más estudiado ya que posee 88 publicaciones que se encargan de una sola función de optimización, sumado a esto 78 publicaciones se han dedicado a la optimización multi-objetivo. La optimización Multi-Objetivo se ha basado en la optimización de makespan con criterios basados en el uso de la maquinaria.

Como se evidencia en este estado del arte, el makespan ha sido el criterio más estudiado y aunque en la presente asistencia de investigación será analizado, la contribución radica en la forma en que se transforma un algoritmo general para criterios regulares diferentes al *makespan* a un caso particular obteniendo soluciones de calidad.

1.6 Modelo de programación lineal para el FJSP

El problema objeto de estudio de la presente asistencia de investigación puede ser modelado como un problema de programación lineal. Como se mencionó en los apartados anteriores, este es un problema complejo y no resulta eficiente emplear la programación lineal en problemas con gran número de variables. No obstante, se solucionarán en instancias con bajo número de máquinas, de operaciones y de pedidos. Este problema es modelado como sigue:

Conjuntos:

- Maquinas (n): Represa el conjunto de las maquinas $\{k = 1,2,3 \dots n)$
- Trabajos (J): Represa el conjunto de las trabajos $\{j = 1,2,3 \dots m)$
- Operaciones (O): Represa el conjunto de operaciones en cada trabajo $O \in J$, $\forall\ Operaciones\ (O)\ en\ los\ trabajos\ J$: $\{i = 1,2,3 \dots p\}$

Parámetros:

- Tiempo de procesamiento: $TP\{\ j\ en\ trabajos, i\ en\ operaciones, k\ in\ maquinas\}$, define el tiempo de procesamiento del trabajo j en la operación i de la maquina asignada k.
- La operación i en el trabajo j *(O):* $O_{i,j}$

Variables:

- Maquina asignada a la operación $\left(MA_{j,i,k} = [0,1]\right), Variable\ Binaria$
- Tiempo de finalización de la operación *(C):* $C_{j,i}$
- $BO_{j,i,j',i',k}$: $1\ si\ la\ operacion\ O_{j,i}\ es\ realizada\ antes\ que\ la\ operacion\ O_{j',i'}\ , en\ la\ maquina\ k$ variable binaria.
- $AO_{j,i,j',i',k}$: $1\ si\ la\ operacion\ O_{j,i}\ es\ realizada\ despues\ que\ la\ operacion\ O_{j',i'}\ , en\ la\ maquina\ k$ variable binaria.

Restricciones

- $\forall\ \{j\ en\ trabajos, i\ in\ operaciones\}$: $\sum_1^k MA_{j,i,k} = 1$, esta restricción permite que solo una maquina sea asignada en cada operación.

- $\forall\ \{k\ en\ maquinas, j\ en\ trabajos, i\ in\ operaciones, j'en\ trabajos, i'in\ operaciones: j \neq j', i \neq i'\}$: $AO_{j,i,j',i',k} = BO_{j',i',j,i,k}$, esta restricción permite definir que la operación (j',i') que precede a una operación (j, i) , tenga como sucesor las misma operación (j,i).

- $\forall \{j\ en\ trabajos, i\ in\ operaciones\}: C_{j,i} = Max\left(C_{j,i-1}; \{k\ in\ maquinas, j'en\ trabajos, i'in\ operaciones\} C_{j',i'} * BO_{j',i',j,i,k}\right) + TP_{j,i,k} * MA_{j,i,k}$.

Esta restricción permite calcular el tiempo de finalización de cada operación teniendo en cuenta el máximo tiempo de finalización de las operaciones que lo anteceden por trabajo y por máquina.

Función Objetivo:

$$Minimizar: \{j\ en\ trabajos, i\ en\ operaciones\} Max(\ C_{j,i})$$

Esta función objetivo busca minimizar el tiempo de finalización entre todos los trabajos realizados.

CAPÍTULO 2. FORMULACIÓN DE LA HEURÍSTICA

Las heurísticas son metodologías que permiten definir una solución factible teniendo en cuenta una serie de consideraciones que permiten acercarse en la mayor medida posibles a la solución óptima de un problema (Correa, Rodríguez & Londoño M., 2008). Por consiguiente la solución que se obtenga es considerada de calidad. Esta solución debe lograrse en tiempos computacionales reducidos.

En este capítulo, se detalla la heurística que soluciona el problema de la siguiente forma: Primero se detallan los pasos para obtener la solución inicial. Seguidamente la estructura de vecindario, las expresiones para determinar el valor del criterio al trasladar una operación crítica y finalmente los pasos.

2.1 Solución inicial para el FJSP

Como se mencionó en el primer capítulo, FJSP soluciona dos problemas (asignación y secuenciación). Estos problemas junto con algunas reglas de ordenamiento se realizan así:

Para la asignación de máquinas a las operaciones, se introduce la variable carga para cada una de las máquinas factibles, la cual representa el tiempo acumulado de utilización de la maquinaria. Inicialmente todas las máquinas tienen carga de cero (ninguna operación ha sido asignada). Cuando se va asignar una máquina a una operación se determina el tiempo de inicio de la operación al comparar el tiempo de finalización de su predecesor por máquina y por pedido y luego se le suma el tiempo de operación en la máquina. La máquina que se asigne será aquella que genere menor tiempo de finalización.

El procedimiento se describe a continuación:

1. Los *n* pedidos son ordenados según su número de identificación de manera ascendente. Es decir pedido 1, pedido 2,… pedido n-esimo.
2. En cada pedido, se recorre la ruta de las operaciones y se va asignando según la máquina que genere la menor carga. Simultáneamente, en la máquina asigna se

actualiza la carga. Es importante aclarar que en caso de que la operación solo exista una máquina para realizar la operación, la heurística asigna la operación y se actualiza la carga.

Una vez las operaciones han sido asignadas, la regla de secuenciación es como sigue:

1. Ordenar de manera creciente los pedidos según el tiempo de procesamiento. Donde el tiempo de procesamiento se calcula al sumar los tiempos de procesamiento de las máquinas en el que fueron asignadas las operaciones.
2. Una vez los pedidos estén ordenados se toman las i-esimas operaciones sobre la ruta de cada pedido y se secuencian de manera ascendente según el tiempo de operación. Este proceso finaliza cuando se ha asignado la última operación del pedido con el mayor número de operaciones.
3. Los arcos conjuntivos se van agregando en la medida en que se van secuenciando las operaciones en las máquinas y vale la pena resaltar que el tiempo de inicio para una operación será el máximo entre la finalización de su predecesor inmediato por máquina y sobre la ruta del pedido.

2.2 Estructuras de vecindarios

Los experimentos que se presentan están soportados en dos vecindarios denominados N_1 y N_2. Un ejemplo de estos vecindarios se ilustra en la Figura 1, en donde la ruta crítica es señalada por líneas discontinua. N_1 (grafo de la izquierda) está formado por todas las operaciones críticas (operaciones sombreadas) que afectan un criterio, por ejemplo las operaciones críticas del pedido de mayor tiempo de finalización para el caso de *makespan*. Mientras que N_2 (grafo de la derecha) solo selecciona aquellas que pertenecen a arcos críticos. Por consiguiente un arco crítico está formado por dos operaciones críticas consecutivas que se procesan sobre la misma máquina y que hacen parte de la ruta de dos pedidos diferentes.

Nótese, que $N_2 \subset N_1$. En la Figura 1, las operaciones {1, 2, 4 y 8} pertenecen a N_1, mientras que las operaciones {2, 4 y 8} pertenecen a N_2. En los experimentos se validarán si con menor esfuerzo computacional N_2 conduce a mejores soluciones.

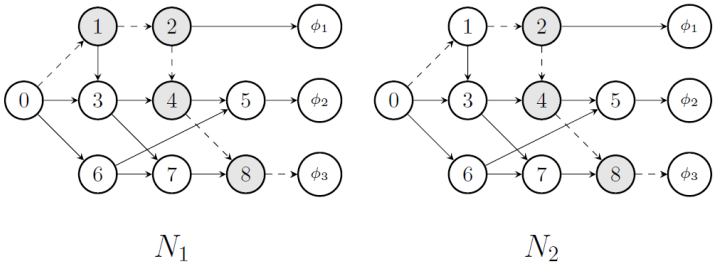

Figura 2. *Ejemplo de vecindarios para el FJSP*

2.3 Factibilidad del movimiento de operaciones críticas

Para la factibilidad del movimiento de operaciones críticas se han considerado las condiciones A del test HTA propuesto por (García-León., 2016). Estas condiciones parten del concepto del número máximo de arcos de nodo y para que no exista una ruta, se explica con dos nodos u y v. En este sentido no existe una ruta entre u y v si el número de arcos de u es mayor. Esta condición se aplicara al reasignar una operación crítica x al reasignarse entre dos operaciones j y k. Es decir se debe asegurar que no existe una ruta entre el sucesor en ruta de x y J, ni entre k y el predecesor en ruta de x. La Figura 2 muestra la transformación en el grafo para lograr dicho propósito sin crear un ciclo, Frx representa el sucesor inmediato en secuencia de x y prx representa el predecesor. Así mismo se ilustra el cambio en los arcos.

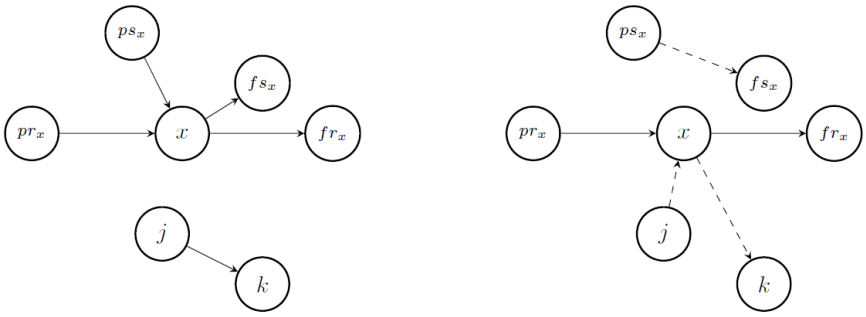

Figura 3. *Inserción de x entre j y k*

2.4 Funciones de estimación

El propósito de las funciones consiste en determinar el valor del *makespan* sin hacer cambios en el grafo actual cuando el arco crítico (x,y) se invierte o para la reasignación de x e y entre j y k. Las funciones de estimación para la inversión de arcos críticos son extraídas de (Mati et al., 2011) la cual considera la inversión de arcos críticos en el problema clásico de Job-shop. Con respecto a la reasignación se propone la siguiente expresión

$$\hat{C}_i = \begin{cases} C_i & \text{si } q_\epsilon^i = q_k^i = -\infty \\ \max\{L_1, L_2\} & \end{cases}$$

Con

$$L_1 = h_{P_{sx}} + p_{P_{sx}} + p_{F_{sx}} + q_{F_{sx}}$$
$$L_2 = \hat{h}_x + p'_x + \max\{p_k + q_k, p_{F_{rx}} + q_{F_{rx}}\}$$

Donde p'_x es el tiempo de procesamiento de x luego del movimiento y

$$\hat{h}_x = \max\{h_j + p_j, h_{P_{rx}} + p_{rx}\}$$

2.5 Funcionamiento de la heurística

La heurística se estructura en la aplicación secuencial de dos etapas: Mejoramiento y diversificación. En la etapa de mejoramiento, se estima el criterio al invertir un arco crítico (x,y) y la reasignación de x y y según la condición de factibilidad. En esta etapa se selecciona el mejor movimiento para generar una nueva solución, la cual se debe asegurar que se genere un menor valor para el *makespan*.

Cuando no sea factible mejorar el valor del criterio, la heurística realiza una diversificación la cual consiste en generar un movimiento aleatorio durante t iteraciones. Este valor de t, está comprendido entre [a,b]. Si el valor del criterio no logra ser minimizado un nuevo proceso de búsqueda local inicia en el cual el mejor valor del criterio es almacenado y se reinicia el proceso optimizando la variable local no la general.

CAPÍTULO 3. EXPERIMENTOS COMPUTACIONALES

Este capítulo explica el cumplimiento de los objetivos que se trazaron para la presente asistencia de investigación**.** Los experimentos se realizaron así: Primero se halla la solución óptima del *makespan* para instancias de tamaño pequeño; seguidamente se detalla un ejemplo de solución inicial, luego el proceso de calibración de los parámetros [a,b] durante el proceso de diversificación en la búsqueda local y finalmente se halla el *makespan* al contrastar los resultados con la solución óptima en instancias pequeñas y con respecto a los mejores valores conocidos.

3.1 Solución óptima al FJSP al minimizar *makespan*

Antes de empezar los procesos experimentales, se define la concepción matemática del *makespan*, el cual es basado en un conjunto J de n trabajos $J = \{J_1, J_2, \ldots, J_n\}$ y en C_i el tiempo de finalización de J_i . Por consiguiente *makespan* $= C_{max} = \max C_i \; \forall \; i \; \in J$.

Es difícil determinar instancias de tamaño pequeño para optimizar el *makespan*. Sin embargo, es importante resaltar que por ser un problema complejo no se obtendrá la solución con eficiencia computacional. Las instancias a considerar provienen de (Hurink et al., 1994) para problemas de cinco máquinas con diez pedidos y 50 operaciones y se encuentran denotadas de la01 a la05 en tres niveles de flexibilidad que varían según el número promedio de máquinas disponibles para procesar una operación en "*e-data, r-data y v-data*". El problema se codifico en el software GUSEK y en la Tabla 1 se registraron los resultados como también el tiempo en que se obtiene la solución óptima y comparando con los mejores valores conocidos.

En la Tabla 1, la columna Mejor valor conocido representa el mejor valor que se ha encontrado en la literatura científica. Columna Sol óptima es la solución óptima que determina el software y la columna Tiempo (s) determina el tiempo empleado por el software en segundos para determinar la solución óptima. Fíjese que se están considerando tres conjuntos en donde *e-data* representa una configuración clásica de *Job-shop* y en algunos casos *v-data* tiende a representar un open shop; es decir una operación se puede procesar en cualquier máquina.

Si se opta por tomar una decisión optima, el tiempo de espera es considerablemente largo. Por ejemplo en v-data en La05 habría que esperar más de 10 horas (36000 segundos). Lo anterior refleja la necesidad del desarrollo de heurísticas que mitiguen este impacto.

Tabla 1. Solución óptima para instancias de 5 x 10 de (Hurink et al., 1994)

Instancia	e-data			r-data			v-data		
	Mejor valor conocido	Sol óptima	Tiempo (s)	Mejor valor conocido	Sol óptima	Tiempo (s)	Mejor valor conocido	Sol óptima	Tiempo (s)
La01	609	609	13896,4	570	570	15247,1	570	570	17754,9
La02	655	655	16745,2	529	529	19254,6	529	529	24368,2
La03	563	550	17598,2	478	477	24562,3	479	477	29358,4
La04	570	568	21325,7	507	502	28954,3	502	502	35258,4
La05	503	503	14267,4	457	457	25456,2	457	458	36985,2

3.2 Ejemplo de solución inicial

Para este apartado se considera la instancia ilustrada en la Figura 1. Los pasos que se siguen se describen a continuación.

Primero se procede a crear las variables de carga, que representan el tiempo total que ha sido asignado a cada una de las maquinas. Se recorre el grafo y se calcula la carga para aquellas maquinas que se encuentran en una operación que únicamente se puede realizar en una maquina (Figura 4). Nótese, que en la parte superior izquierda se ha considerado los tiempos de carga.

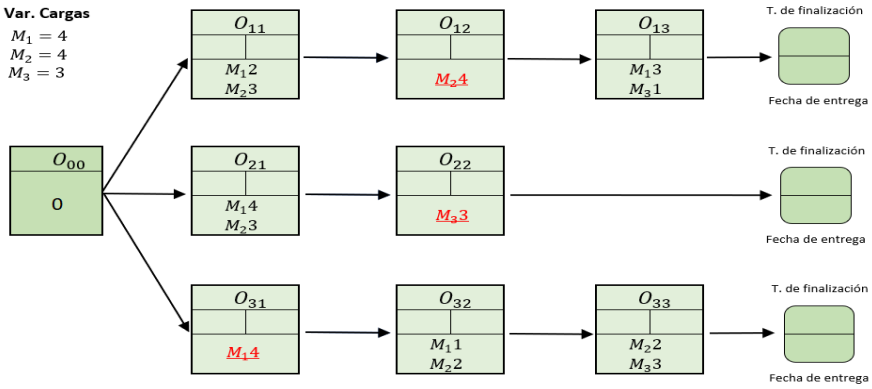

Figura 4. Asignación de cargas con operaciones de una sola maquina

Posteriormente se recorre el grafo asignado en cada operación la maquina con menor carga dentro de las maquinas posibles de cada una y se actualiza la carga de la maquina asignada (Figura 5).

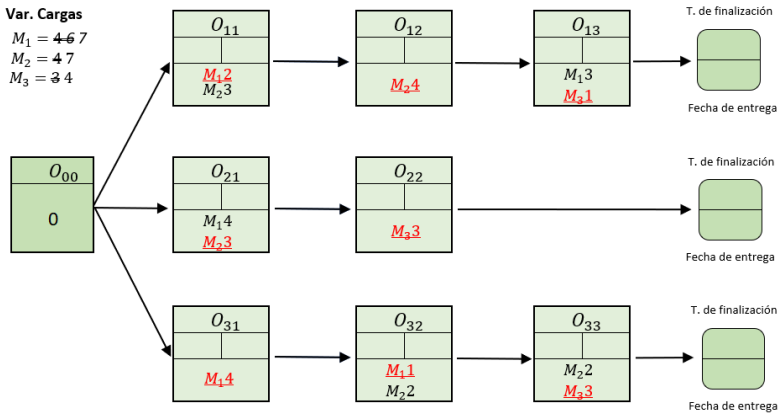

Figura 5. Asignación de máquinas a cada operación

Teniendo la asignación completa se procede a realizar la secuenciación por máquinas, esta requiere de identificar el tiempo de procesamiento de cada pedido (suma de los tiempos de procesamiento de cada operación en el pedido según la maquina asignada). Para el primer pedido se tiene un tiempo de 7, para el segundo pedido 6 y para el tercero 8. Se procede por maquina a recorrer el grafo enrutando la secuencia de esta teniendo en cuenta de menor a mayor el tiempo de procesamiento (ver Figura 6).

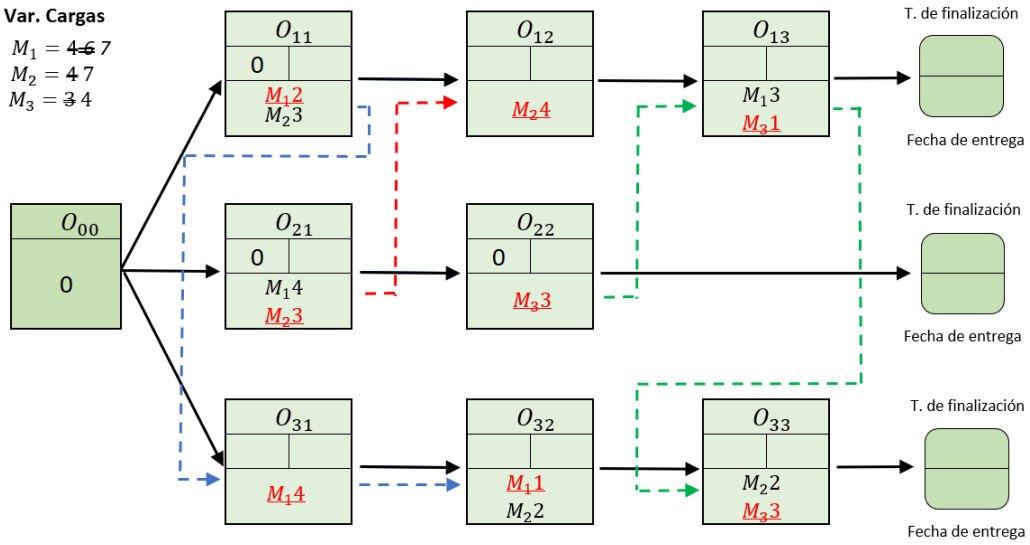

Figura 6. Asignación y secuenciación completa

Finalmente se da solución al grafo, teniendo en cuenta que una operación no podrá ser interrumpida ni podrá dar comienzo si su antecesor por trabajo y por maquina no ha finalizado, dicha solución se presenta a continuación (Figura 7).

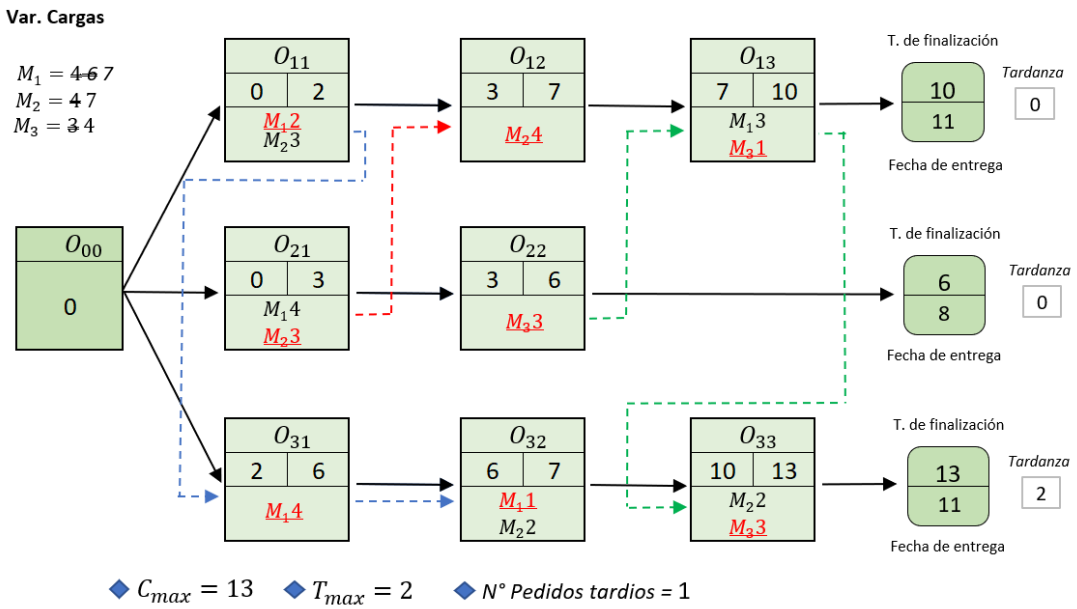

Figura 7. Grafo de la solución inicial del modelo de JSF

Los resultados de algunos criterios de la solución inicial de este ejemplo corresponden a: un makespan ($C_{\max}$) de 13, una tardanza máxima ($T_{\max}$) de 2, y un número total de pedidos tardíos de 1.

3.3 Calibración de parámetros a y b durante el proceso de diversificación

Antes de empezar a ejecutar la heurística para las instancias más conocidas en la literatura, se han considerado las 18 instancias DPP desde *DPP01* hasta *DPP18* de (Dauzère-Pérès & Pauli, 1998). Estas instancias son las más complicadas que existen y la preocupación de los enfoques centrados en el *makespan* es mejorar el valor conocido. En la asistencia se han estas instancias y encontrar los parámetros de a y b que en promedio se aproximan más al mejor conocido al analizar las dos estructuras de vecindario N_1 y N_2.

Durante el periodo de la asistencia de investigación se hicieron diferentes corridas de experimentos con las instancias. Sin embargo, en la Tabla 2, se registraron los parámetros que mejor calibraron la cercanía del valor del criterio con el mejor conocido. En este caso se logra cuando el intervalo esta entre [4,8]. Con estos parámetros se procede a

realizar la experimentación. En la Tabla 2, se ilustran los resultados que se obtienen en un experimento de 7200 segundos por instancia y por vecindario (N_1 y N_2). Esto no significa que el resultado se logra en este tiempo, este es el tiempo de ejecución. La columna de ***Mejor valor conocido*** representa el mejor valor hallado para el makespan en la instancia (no el óptimo); esta información fue extraída de (González et al., 2015). De nuevo la columna *F* representa la disponibilidad promedio para procesar una operación; por ejemplo en DPP01 1,13, significa que en promedio existe 1.13 máquinas para que la operación sea realizada.

A partir de la Tabla 2 se puede inferir que para altos valores de *Flexibilidad* se requerirá de mayor esfuerzo computacional al evaluar en diferentes máquinas la estimación del movimiento y que para bajos valores se asemeja al clásico *Job-shop*. Así mismo el mejor desempeño se alcanza cuando el parámetro a toma un valor de 4 y b de 8 y la evidencia de ello se nota al subrayar los valores del criterio en la Tabla. Si se analiza la dominancia entre N_1 y N_2 no es posible determinar una dominancia representativa. Sin embargo, es importante resaltar que N_2 requiere menor esfuerzo computacional y que obtiene mejores resultados en 10 de 18 instancias.

Tabla 2. *Calibración de parámetros [a,b] en instancias de (Dauzere-Peres & Pauli., 1998)*

Instancias	***Mejor valor conocido***	***F***	***[2,5]***		***[2,6]***		***[3,5]***		***[4,8]***	
			N1	***N2***	***N1***	***N2***	***N1***	***N2***	***N1***	***N2***
DPP01	2505	1,13	2559	2547	2554	2554	2558	2554	2525	**2511**
DPP02	2229	1,69	2255	2251	2260	2255	2262	2256	**2241**	2246
DPP03	2228	2,56	2254	2233	2246	2243	2253	2242	2234	**2233**
DPP04	2503	1,13	2542	2534	2552	2541	2549	2543	**2506**	2511
DPP05	2211	1,69	2259	2256	2258	2253	2261	2260	**2360**	2386
DPP06	2183	2,56	2236	2234	2232	2232	2232	2236	**2219**	2220
DPP07	2266	1,24	2420	2540	2429	2534	2426	2541	**2360**	2386
DPP08	2064	2,42	2132	2112	2132	2117	2134	2120	2107	**2102**
DPP09	2062	4,03	2125	2116	2118	2107	2125	2112	2092	**2089**
DPP10	2267	1,24	2527	2894	2530	2889	2534	2890	2376	**2374**
DPP11	2051	2,42	2147	2144	2141	2144	2148	2145	2108	**2104**
DPP12	2018	4,03	2118	2191	2108	2200	2114	2201	2074	**2068**
DPP13	2248	1,34	2442	2440	2441	2433	2445	2437	2348	**2345**
DPP14	2163	2,99	2255	2408	2258	2400	2256	2408	**2208**	2210
DPP15	2162	5,02	2233	2242	2235	2235	2238	2244	**2197**	2200
DPP16	2244	1,34	2476	3075	2483	3084	2483	3080	2347	**2330**
DPP17	2130	2,99	2239	2424	2230	2433	2239	2432	2194	**2190**
DPP18	2119	5,02	2206	2218	2208	2212	2208	2218	**2182**	2186

3.4 Cálculo del *makespan* para el FJSP

Luego de que se han calibrado los parámetros [a,b] se procede a ejecutar la heurística en la herramienta computacional que ha sido diseñada. Esta herramienta fue diseñada en el lenguaje *Java* y en el ambiente de desarrollo *Net-beans 8.0*. En este apartado se realizan los cálculos para las instancias analizadas previamente.

El experimento consistió en realizar diez experimentos por instancia y por vecindario con el fin de determinar la estabilidad en los resultados al registrar el mínimo valor, el máximo valor y la desviación estándar "σ" y los resultados se registran en la Tabla 3.

***Tabla 3**. Cálculo del* makespan *para las instancias de (Dauzere-Peres & Pauli., 1998)*

Instancias	*F*	*Mejor valor conocido*	*Mejor resultado*	*Error (%)*	*N1*			*N2*		
					Min	*Max*	*σ*	*Min*	*Max*	*σ*
DPP01	1,13	2505	2511	0,2	2525	2564	13,3	**2511**	2554	14,7
DPP02	1,69	2229	2241	0,5	**2241**	2258	4,6	2246	2257	3,4
DPP03	2,56	2228	2233	0,2	2234	2242	2,3	**2233**	2241	2,3
DPP04	1,13	2503	2506	0,1	**2506**	2553	14,9	2511	2546	11,8
DPP05	1,69	2211	2360	6,7	**2360**	2248	4,4	2386	2245	3,4
DPP06	2,56	2183	2219	1,6	**2219**	2227	2,4	2220	2227	2,7
DPP07	1,24	2266	2360	4,1	**2360**	2448	18,3	2386	2443	21,2
DPP08	2,42	2064	2102	1,8	2107	2121	4,5	**2102**	2118	5,9
DPP09	4,03	2062	2089	1,3	2092	2105	4,3	**2089**	2102	4,2
DPP10	1,24	2267	2374	4,7	2376	2454	22,1	**2374**	2455	21
DPP11	2,42	2051	2104	2,6	2108	2120	4,7	**2104**	2123	6,4
DPP12	4,03	2018	2068	2,5	2074	2083	3,5	**2068**	2083	4,9
DPP13	1,34	2248	2345	4,3	2348	2382	10,5	**2345**	2397	16,9
DPP14	2,99	2163	2208	2,1	**2208**	2217	3,3	2210	2221	5,3
DPP15	5,02	2162	2197	1,6	**2197**	2214	5,8	2200	2212	4,5
DPP16	1,34	2244	2330	3,8	2347	2383	12,2	**2330**	2344	5,3
DPP17	2,99	2130	2190	2,8	2194	2201	2,8	**2190**	2208	7,7
DPP18	5,02	2119	2182	3,0	**2182**	2196	5,3	2186	2204	6,8

En la Tabla 3 se presentan los resultados de cada vecindario y en cada uno de ellos se ha registrado el mínimo valor encontrado (columna *Min*), el máximo valor (columna *Max*) y su respectiva desviación (columna *σ*). Adicional a las columnas de la Tabla 2, se ha adicionado las columnas *Mejor resultado* el cual el mejor valor hallado en los experimentos

y la columna *Error*, el cual representa el error porcentual del mejor resultado respecto al mejor valor conocido. Si se tratará de condiciones ideales σ debería ser cero (0) si se hablará de una heurística estable. Sin embargo, por tratarse unidades de miles σ no es superior a 21,2 unidades (instancia *DPP07* en N_2). Y para la mayoría de las otras instancias están por debajo de 10 unidades, lo cual evidencia la validez de los análisis de los experimentos.

De acuerdo con lo anterior, las dos estructuras de vecindarios generan resultados homogéneos y esto se debe en parte a que son algoritmos probabilísticos; por ejemplo al seleccionar un valor aleatorio en el intervalo [a,b]. Si se analiza la calidad de los resultados, el porcentaje de error no es superior al 6,7% (Instancia *DPP05*). Para las instancias de baja flexibilidad este porcentaje está por debajo del 1%. Por ejemplo instancias *DPP01*, *DPP02* y *DPP04*. No obstante, en las instancias *DPP05*, *DPP10*, *DPP13* y *DPP16*. No se presenta este comportamiento, lo que motiva a la revisión o a la formulación de estrategias de diversificación para refinar el proceso de búsqueda.

3.5 Comparación de la heurística con la programación lineal

Una vez se han validado los resultados de la heurística en las instancias de gran tamaño, se procede a calcular el *makespan* en las instancias de (Hurink et al., 1994) para evaluar la distancia respecto a la mejor conocida. Para este experimento se ejecutan 10 corridas de la heurística durante un lapso de 300 segundos considerando el vecindario N2 el cual condujo a los mejores resultados. Los resultados se registraron en la Tabla 4 al considerar el mínimo (columna *Mínimo*), el promedio (columna *Media*) y el procentaje de error del promedio respecto al mejor valor conocido (Columna *Per*) y considerando los tres conjuntos de datos según la flexibilidad (*e-data*, *r-data* y *v-data*).

Tabla 4. Comparativo de la solución óptima y la heurística en la instancias de (Hurink et al., 1994)

Instancia	*Mejor valor conocido*	*e-data*			*Mejor valor conocido*	*r-data*			*Mejor valor conocido*	*v-data*		
		Mínimo	*Media*	*Per (%)*		*Mínimo*	*Media*	*Per (%)*		*Mínimo*	*Media*	*Per (%)*
La01	609	**609**	612,4	0,6	570	572	576,4	1,1	570	**570**	578,2	1,4
La02	655	**655**	661,4	1,0	529	531	537,6	1,6	529	**529**	533,4	0,8
La03	550	554	557,2	1,3	477	478	484,6	1,6	477	479	486,1	1,9

La04	568	**568**	570,4	0,4	502	504	507,1	1,0	502	**502**	505,9	0,8
La05	503	**503**	505,8	0,6	457	458	462,3	1,2	457	458	463,7	1,5

De la Tabla 4, se puede inferir que la heurística alcanza el mejor valor conocido en cuatro de cinco instancias para e-data y en tres de cinco instancias en v-data. Aunque en r-data no se alcanza el porcentaje de error entre la media y el mejor valor conocido no es superior al 1,6%. Respecto al mínimo valor encontrado la diferencia no es superior a dos unidades de tiempo.

Los resultados evidencian, que aunque la heurística está formulada para optimizar criterios regulares diferentes al *makespan*. Sin emabrgo al optimizar el criterio se alcanzan resultados de calidad en tiempos computacionales razonables. No obstante, si la estrategia a seguir es calibrando el algoritmo mediante el mejoramiento en la etapa de diversificación diferentes estrategias pueden considerarsen como: Continuar calibrando los límites del intervalo [a,b] u optimizar un criterio auxiliar que en este caso podría ser la suma de los tiempos de finalización de los pedidos.

CONCLUSIONES

En la presente asistencia de investigación se ha formulado una heurística para solucionar el problema de *Scheduling Job-Shop* flexible "FJSP" al minimizar el *makespan*. La heurística se presenta una herramienta computacional cuyo ingreso es un archivo de texto con la información del problema y la salida que genera es un archivo con la solución optimizada del problema. El diseño de la herramienta computacional satisface tres objetivos específicos:

El primero es formular funciones de estimación para el movimiento de operaciones críticas el cual se cumple al formular un test de factibilidad basado en el concepto de orden de operaciones o de máximo número de arcos, la inversión de arcos críticos con una función de estimación para la reasignación de operaciones. Así mismo se analizaron dos estructuras de vecindarios (N_1 y N_2). En los experimentos computacionales N_2 es superior a N_1.

El segundo objetivo es aplicar el modelo de Programación Lineal en instancias de tamaño pequeño. Su cumplimiento se genera al considerar instancias de (Hurink et al., 1994) y se evidenció tiempos computacionales polinomiales y algunos de ellos requirieron de 10 horas para determinar la solución óptima. El fin de la heurística consistió en llegar a soluciones óptimas y cercanas a ellas en solo 300 segundos de tiempo computacional.

El tercer objetivo es la formulación de un modelo de búsqueda local. Este modelo se formula en dos etapas: Mejoramiento y diversificación. En mejoramiento se selecciona la solución que minimiza el criterio y en la diversificación un número de movimientos aleatorios para escapar de un óptimo local. En el caso particular del makespan, se encontró que un número entre [4,8] conduce el proceso a regiones de búsqueda inexplorables.

Esta asistencia contribuye al proyecto con:

Generación de nuevo conocimiento al proponer un algoritmo de solución inicial al introducir el concepto de carga de máquinas. Los experimentos computacionales contribuyeron a demostrar la eficiencia del algoritmo para una publicación de cuartil Q1.

REFERENCIAS BIBLIOGRÁFICAS

[Brucker, 2007] Brucker, P. (2007). Scheduling algorithms. Springer.

[Brucker and Schlie, 1990] Brucker, P. and Schlie, R. (1990). Job-shop scheduling with multi-purpose machines. Computing, 45(4):369–375.

[Correa A, Rodriquez E. & Londoño M., 2008] Correa A, Rodriquez E. & Londoño M., 22 diciembre-2012. Título: ”Scheduling for shop settings types to flexible Job Shop:State of the art”. Pg. 1-5, 7-10.

[Chaudhry and Khan, 2015] Chaudhry, I. A. and Khan, A. A. (2015). A research survey: review of flexible job shop scheduling techniques. International Transactions in Operational Research.

[Dauzère-Pérès et al., 1998] Dauzère-Pérès, S., Roux, W., and Lasserre, J. (1998). Multi-resourceshopschedulingwithresourceflexibility. EuropeanJournalofOperational Research, 107(2):289–305.

[Garcia Leon A., 2016] Garcia Leon A. (10/05/2016). Título: “Développement et validation de nouvelles approches d'optimisation pour le problème d'ordonnancement de type job shop flexible”.

[Hurink et al., 1994] Hurink, J., Jurisch, B., and Thole, M. (1994). Tabu search for the job-shop scheduling problem with multi-purpose machines. OR Spectrum, 15(4):205–215.

[Jurisch, 1995] Jurisch, B. (1995). Lower bounds for the job-shop scheduling problema on multi-purpose machines. Discrete Applied Mathematics, 58(2):145–156.

[Mati and Xie, 2004] Mati, Y. and Xie, X. (2004). The complexity of two-job shop problems with multi-purpose unrelated machines. European Journal of Operational Research, 152(1):159–169.

[Motoa T. & Osorio J., 2012] Motoa T. & Osorio J. (junio, 2012).Titulo: *"*Hierarchical production planning in the flexible job shop

[Tejada G., 2016]Tejada G. (09/11/2016).Titulo: "Entrutamiento y secuenciación optimos en un Flexible Job Shop multiobjetivo"- UNMSM.