MINIMIZACIÓN DEL MAKESPAN EN EL PROBLEMA DE JOB SHOP FLEXIBLE CON RESTRICCIONES DE TRANSPORTE UTILIZANDO ALGORITMO GENÉTICO

JUAN DAVID GÓMEZ MORENO
EDWIN ALFREDO ORDUZ GONZÁLEZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍA FÍSICO MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2015

MINIMIZACIÓN DEL MAKESPAN EN EL PROBLEMA DE JOB SHOP FLEXIBLE CON RESTRICCIONES DE TRANSPORTE UTILIZANDO ALGORITMO GENÉTICO

JUAN DAVID GÓMEZ MORENO
EDWIN ALFREDO ORDUZ GONZÁLEZ

Trabajo de grado para optar al título de Ingeniero Industrial

Director:
CARLOS EDUARDO DÍAZ BOHÓRQUEZ
Magister en Ingeniería Industrial

Codirector:
MYRIAM LEONOR NIÑO LÓPEZ
Doctora en dirección y organización de empresas

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍA FÍSICO MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2015

A Dios, por permitirme vivir esta experiencia y guiar mi vida durante estos años. Es él quien ha llenado mi vida de personas especiales y retos importantes.

A mis padres, hermanos y sobrinos, quienes han sido el motor para cumplir un logro más en mi vida profesional, brindándome su apoyo, consejo y comprensión. Esto es por ustedes y para ustedes.

A María Camila, quien se ha convertido en esa persona especial que me brinda su amor y apoyo. Un logro más cumplido que nos acerca a nuestros sueños.

Edwin Alfredo

En primera instancia a Dios por ser el guía en cada uno de mis pasos y por bríndame una vida llena de aprendizajes, experiencias y sobre todo felicidad.

A mis padres, Miguel Gómez y Gloria Moreno, los cuales han sido mi fuente de inspiración y motivo de orgullo en quienes siempre encontré apoyo incondicional para este logro. Para ustedes todo.

A Jose M. Gómez, el mejor hermano y modelo a seguir por sus consejos y acompañamiento a lo largo de toda mi carrera profesional.

A Laura Contreras, por ser mi fortaleza en los momentos de debilidad y por su amor incondicional.

Juan David

AGRADECIMIENTOS

A nuestro mentor, MSc. Carlos Eduardo Díaz, por ser nuestro guía durante la formación profesional recibida. A la Dra. Myriam Leonor Niño por la asesoría brindada durante esta etapa de nuestra carrera profesional.

A la Universidad Industrial de Santander, Escuela de Estudios Industriales y Empresariales y al Grupo OPALO por su aporte a nuestra formación en el crecimiento personal y profesional.

A nuestros amigos y demás familiares, quienes nos acompañaron durante este importante trayecto y de quienes tendremos los mejores recuerdos.
Infinitas gracias a todos.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

RESUMEN

TÍTULO: MINIMIZACIÓN DEL MAKESPAN EN EL PROBLEMA DE JOB SHOP FLEXIBLE CON RESTRICCIONES DE TRANSPORTE UTILIZANDO ALGORITMO GENÉTICO[1]

AUTORES: GÓMEZ MORENO, Juan David. ORDUZ GONZALEZ, Edwin Alfredo.[2]

PALABRAS CLAVE: Algoritmo Genético; Job Shop Flexible; Metaheurística; Restricciones de Transporte.

DESCRIPCIÓN

La presente investigación aborda el problema de secuenciación y asignación de máquinas *Flexible Job Shop Scheduling* (FJSSP) con restricciones de transporte en búsqueda de la minimización del *Makespan* como función objetivo. El FJSSP es considerado un problema de optimización combinatoria de tipo *NP-Hard* por su complejidad computacional y es de gran importancia en la industria por la optimización de recursos que representa. La técnica a utilizar para solucionar el problema es el algoritmo genético como adaptación de la evolución biológica a la inteligencia artificial el cual se ajusta a las características del mismo. El Algoritmo Genético propuesto utiliza los operadores de selección por torneo, de cruce de un punto y de mutación SWAP basado en la estrategia de Zhang. *et al* (2011)[3].

Se realizó una validación a través de la comparación de los resultados obtenidos versus las instancias de diferentes etapas representadas por medio de la descomposición del Job Shop Flexible con recursos de transporte de manera progresiva, donde se comprobó la eficiencia y eficacia del algoritmo propuesto.

Finalmente, con el fin de identificar la influencia de cada factor en la función objetivo, se realizó un diseño de experimentos 2^3 con 7 instancias diseñadas y desarrolladas por otros autores reconocidos en el mundo de la investigación.

[1] Trabajo de grado

[2] Facultad de Ingenierías Físicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: MSc. Carlos Eduardo Díaz Bohórquez

[3] ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, 2012, vol. 39, no 7, p. 1713-1723.

ABSTRACT

TITLE: A GENETIC ALGORITHM FOR MINIMIZING MAKESPAN IN FLEXIBLE JOB SHOP PROBLEM WITH TRANSPORT CONSTRAINS[4]

AUTHORS: GÓMEZ MORENO, Juan David. ORDUZ GONZALEZ, Edwin Alfredo.[5]

KEY WORD: Genetic Algorith; Flexible Job Shop; Metaheuristics; Transport Constrains.

DESCRIPTION:

This research addresses the problem of sequencing machines and allocation of Flexible Job Shop Scheduling (FJSSP) with transport constrains in pursuit of minimizing the makespan. The FJSSP is considered a combinatorial optimization problem NP-Hard type because of its computational complexity and represents great importance in the industry for optimizing resources. The technique used to solve the problem is the genetic algorithm as an adaptation of biological evolution to artificial intelligence which fit the characteristics of the problem. The proposed genetic algorithm uses the tournament selection, crossover and mutation SWAP operators based on the strategy of Zhang. *Et al* (2011)[6].

To test the efficiency and effectiveness of the proposed algorithm, the validation is performed by comparing the results obtained versus different instances of steps represented by the decomposition of Flexible Job Shop transport resources.

In the other hand, in order to identify the effect of each input parameter on the objective function, a design of experiments was carried out with 7 instances designed and developed by other authors recognized in the world of research. The results show that the proposed genetic algorithm is efficient in different configurations of the Classic Job Shop listed above and for the Flexible Job Shop restricted transport having solutions closely approximate to the best found until today.

[4] Research Project.

[5] Facultad de Ingenierías Físicomecánicas. Escuela de Estudios Industriales y Empresariales. Director: MSc. Carlos Eduardo Díaz Bohórquez

[6] ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. *Computers & Operations Research*, 2012, vol. 39, no 7, p. 1713-1723.

INTRODUCCIÓN

En la actualidad, la optimización de los recursos productivos juega un papel muy importante en la planeación de la producción dentro de las organizaciones, puesto que estas se desenvuelven en un mundo competitivo y globalizado, donde se requiere ser eficiente en sus operaciones.

Una adecuada programación dentro del área de producción, que involucre una asignación de máquinas para distintas operaciones y la secuenciación de estas, se ha convertido en un proceso de gran importancia en los sistemas de producción, debido a que aporta múltiples beneficios a las empresas como disminución de costos, mayor aprovechamiento de recursos y mejorar el servicio al cliente.

Debido a esto, en los últimos años investigadores como Zhang (2011), Tang (2011), Gonzalez (2013), Rossi (2014) entre otros, han propuesto y estudiado un problema que describe este tipo de situaciones, el *Job Shop flexible*, problema en el cual se cuenta con un conjunto de trabajos J compuestos por operaciones O_{ij} que deben ser programadas y ejecutadas en un conjunto de máquinas m en determinado periodo de tiempo.

En el proyecto a realizar se pretende trabajar el problema de la programación, asignación y secuenciación de máquinas para la ejecución de múltiples trabajos (*FJSSP*) con restricciones de transporte; un problema de optimización combinatoria

de tipo *NP-Hard*[7] con características adicionales al *Job Shop* clásico, que permite el acercamiento entre la investigación y la realidad de la vida empresarial.

Para lo anterior, se decidió realizar una validación del tema a partir de la revisión bibliográfica con el fin de identificar estrategias tenidas en cuenta a la hora de abordar el problema.

A partir de la revisión, se obtuvo que técnicas exactas no son adecuadas para su solución, por lo que se hizo énfasis en el uso de métodos metaheurísticos. La técnica a utilizar para solucionar el problema son los algoritmos genéticos, establecidos por Holland [8] (1975), quien describe este algoritmo como una adaptación de la evolución biológica a la inteligencia artificial, utilizada ahora en el mundo de la investigación para la resolución de problemas de búsqueda y optimización.

Posteriormente, se construyó un pseudocódigo para abordar el problema del *Job Shop* Flexible con la técnica anteriormente descrita, validando la significancia de los factores requeridos en el método a través de un diseño de experimentos 2^{k-1}, ejecutado a 7 instancias elegidas aleatoriamente.

Finalmente, se presentan las conclusiones obtenidas de la presente investigación y recomendaciones a tener en cuenta para futuras investigaciones en relación al tema abordado.

[7] GAREY, Michael R.; JOHNSON, David S.; SETHI, Ravi. The complexity of flowshop and jobshop scheduling. En: Mathematics of operations research, 1976, vol. 1, no 2, p. 117-129.

[8] HOLLAND, John H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, 1975.

1. PLANTEAMIENTO DEL PROBLEMA

La programación de operaciones ha sido un problema de interés en la investigación, que ha evolucionado de forma específica para poder describir distintos sistemas productivos. A través del tiempo se ha convertido en un proceso de gran importancia para las organizaciones, ya que aporta múltiples beneficios como disminución de costos, mayor aprovechamiento de recursos y mejorar el servicio al cliente.

Durante los últimos años, las investigaciones han avanzado ampliando las diferentes variables que puede tener un sistema productivo, tal y como se observa en el *Job Shop* flexible (FJSSP), tema a tratar, basado en determinar cuál es el recurso más apropiado para ejecutar determinada operación (asignación de recursos) y hallar la programación más eficiente de operaciones dentro de cada una de las maquinas (secuenciación de recursos).

Junto a esto, cuando una máquina finaliza una operación, el trabajo finalizado debe ser transportado a otro lugar, lo que hace necesario contar con recursos para la actividad de transporte, por lo que se incluye dentro del problema un conjunto de recursos idénticos de transporte encargados de tareas de desplazamiento entre máquinas, siendo el tiempo de transporte dependiente de la trayectoria entre las máquinas.

Para problemas de complejidad *NP-Hard*, métodos exactos no son convenientes si se espera obtener una solución aplicable (en términos de tiempo computacionales) y de bajo costo, por lo que se hace necesario el uso de metaheurísticas. El algoritmo genético, es uno de estos métodos, que ha sido empleado para solucionar

problemas de espacio de solución y tiempo computacional de crecimiento exponencial, *NP-Hard*[9].

En la presente investigación, se pretende trabajar el problema del FJSSP con restricciones de transporte, optimizando el *makespan*, por medio de la metaheurística de Algoritmo Genético; validando tanto el modelo como el pseudocódigo por medio de un conjunto de instancias propuestas por distintos autores en la literatura.

[9] JAIN, Anant Singh; MEERAN, Sheik. Deterministic job-shop scheduling: Past, present and future. En: European journal of operational research, 1999, vol. 113, no 2, p. 390-434.

2. JUSTIFICACIÓN DEL PROYECTO

El problema de asignación y secuenciación de máquinas FJSSP, según se expuso anteriormente, es uno de los problemas que se encuentra frecuentemente en distintos ambientes, puesto que las compañías requieren optimizar los recursos productivos con los que cuenta y es por medio de la investigación que se abren nuevos caminos para su mejoramiento, teniendo en cuenta factores tales como el tiempo y el costo de su aplicación tanto en sistemas productivos de bienes y servicios.

Según se observa en la revisión bibliográfica, existen distintos autores que se encuentran desarrollando estudios en esta línea de investigación, la cual ha evolucionado desde su concepción más sencilla (*Job Shop* clásico) hasta instancias donde se incluyen asignación de recursos, tiempos de procesamiento variables, dependencia de tiempos según la secuencia, entre otros.

Por eso, en el grupo OPALO se observa la relevancia de avanzar en el estudio, análisis y solución de este problema, al ser una necesidad real que se presenta dentro de los sistemas productivos de bienes y servicios en las empresas.

En esta investigación se pretende conocer el problema y el funcionamiento del método elegido, para determinar una solución factible y de bajo costo que permita realizar una comparación con respecto a otros artículos similares y en posteriores estudios ser aplicado en ambiente reales.

3. OBJETIVOS

3.1. OBJETIVO GENERAL

Resolver el problema de programación del *Job Shop* flexible (FJSSP) con recursos limitados de transporte por medio de un Algoritmo Genético.

3.2. OBJETIVOS ESPECÍFICOS

- Realizar una revisión de literatura sobre el problema del *Job Shop* flexible así como los enfoques propuestos por distintos autores.
- Diseñar un pseudocódigo en MATLAB del algoritmo genético que sirva para la solución del *Job Shop* con las características nombradas
- Validar el algoritmo a través de las instancias encontradas acerca del problema del *Job Shop* flexible con restricciones de transporte.
- Comparar los resultados obtenidos con los encontrados por otros autores a través de distintos métodos aplicados al problema de optimización abordado.
- Elaborar un artículo académico de carácter publicable que contenga los resultados del proyecto de investigación.

4. MARCO TEÓRICO

4.1. OPTIMIZACIÓN COMBINATORIA

La optimización combinatoria es una rama de la matemática aplicada y la ciencia de la computación la cual se encarga del estudio del modelado y solución algorítmica de problemas donde se busca optimizar una función de variables definidas sobre un conjunto discreto, conocidos en la literatura como problemas de optimización.

Un problema de optimización combinatoria $P(S, f)$ está definido por:

- Conjunto de variables $X = \{x_1, \dots, x_n\}$;
- Dominio de variables $D_1, \dots, D_n$;
- Restricción entre variables
- Función objetivo *f* a minimizar, donde f: $D_1 \times \dots \times D_n \rightarrow \mathbb{R}^+$;

El conjunto de todas las posibles asignaciones factibles es:

$$S = \{s = \{(x_1, v_1), \dots, (x_n, v_n)\} | v_i \in D_i, s \ \ satisface\ todas\ las\ restricciones\}$$

S es el espacio o solución de búsqueda, donde cada elemento del conjunto puede ser considerado como un candidato de solución. Para resolver un problema de optimización combinatoria primero se debe encontrar una solución $s^* \in S$ que dé el valor mínimo de la función objetivo, es decir, $f(s^*) \leq f(s)\ \forall\ s \in S$. s^* se considera como la solución óptima global de (S, f) y el conjunto $S^* \subseteq S$ es considerada como el conjunto de soluciones óptimas globales.[10]

[10] BLUM, Christian; ROLI, Andrea. *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*. En: ACM Computing Surveys (CSUR). 2003, 35(3), 268-308.

Una posible clasificación de los métodos de optimización clásicos podría ser: no lineal, lineal, entera, lineal entera, dinámica, estocástica y multiobjetivo.[11]

Resolver un problema de optimización consiste en encontrar el valor que deben tomar las variables y de esta manera hacer óptima la función objetivo bajo una serie de restricciones establecidas. Estas variables no siempre estarán representadas mediante variables continuas y dependen del tipo de problema. Por esta razón, podemos clasificar los problemas de programación entera según el tipo de variables como programación entera pura (PIP) (variables enteras), programación entera binaria (BIP) (todas son binarias) y programación entera mixta (MIP) (variables enteras, binarias y continuas). [12]

Algunos problemas de optimización clásicos que se encuentran en la literatura pueden ser:

- El problema del agente viajero TSP (*travelling salesman problem*).
- Problemas de asignación.
- Problemas de rutas.
- Problemas de flujo máximo.
- Problema de la mochila.
- Problemas de inventario.

Los problemas de optimización también pueden ser clasificados según su complejidad computacional, es decir, el grado de "dificultad" de desarrollo computacional de los algoritmos de solución del problema.

[11] RAMOS, Andrés, et al.. Modelos Matemáticos de Optimización. Universidad Pontificia Comillas Madrid, 2010.

[12] VIDAL, Aitana. Algoritmos heurísticos en optimización. Tesis de Maestría. Santiago de Compostela: Universidad de Santiago de Compostela. Facultad de Matemáticas. 2013. 94 p.

4.1.1. Tipos de Complejidad Computacional

La teoría de la complejidad estudia los recursos requeridos durante el cálculo para resolver un problema los cuales son el tiempo y el espacio, dados por la cantidad de operaciones que efectúan en el peor caso y en función del tamaño de los datos de entrada respectivamente. Comúnmente la máquina de *Turing* es usada como modelo matemático de un algoritmo y es punto de referencia para la clasificación de los mismos. La clasificación está dada por:

Clase P: Sub conjunto de problemas de la clase *NP* que pueden ser resueltos por un algoritmo en una máquina de *Turing* determinística en tiempo polinomial.

Clase NP: Conjunto de problemas que pueden ser resueltos por un algoritmo en tiempo polinomial por una máquina de *Turing* no determinística. Se caracterizan por la existencia de un algoritmo capaz de verificar su solución. El tiempo de cómputo que se requiere para resolver uno de estos problemas se incrementa conforme crece el tamaño del problema presentando una dependencia funcional tal que no admite ser acotada por un polinomio.[13]

Clase NP completos: Sea P' un problema en la clase NP. Entonces P' es NP-completo si cualquier problema en clase NP se puede reducir a P' en tiempo polinomial.

[13] GAREY, M; GRAHAM R.; ULLMAN J. Worst-case analysis of memory allocation algorithms. Proceedings of the UK Workshop on Computational Intelligence, 1972, p. 143–150.

Clase NP Hard: Subconjunto de la clase NP para los que no se puede tener una solución en tiempo polinomial para todas sus instancias. Todos los problemas de esta clase pueden ser reducidos a NP.[14]

Figura 1. Relación entre los problemas P, NP, NP- completo y NP – duro.

Fuente: Adaptado de DUARTE, Abraham; PANTRIGO, Juan; GALLEGO Micael. Introducción a la optimización. En: Metaheuristicas. Madrid: Dykinson, 2008. p. 1-14.

En cuanto a la complejidad computacional, los problemas de *Shop Scheduling* pertenecen a la clase *NP*. Por lo tanto, el tiempo requerido para resolver la programación de tareas o *Shop Scheduling* se incrementa exponencialmente con el tamaño de las variables del mismo[15].

4.2. SHOP SCHEDULING

Según Brucker (2003), la programación de tareas o *Shop scheduling*, es un problema de optimización combinatoria que describe una gran cantidad de sistemas

[14] CORREA, Alexander A.; RODRIGUEZ, Elkin; LONDOÑO María I., Secuenciación de operaciones para configuraciones de planta tipo flexible Job Shop: Estado del arte. En: Avances en Sistemas e Informática. Diciembre, 2008. vol. 5, no. 3, p. 151-161.

[15] ZOBOLAS, G.; TARANTILIS, C; LOANNOU, G. Exact, Heuristic and Meta-heuristic Algorithms for Solving Shop Scheduling Problems. In Metaheuristics for Scheduling in Industrial and Manufacturing Applications. Springer Berlin Heidelberg. 2008.

productivos, caracterizado por múltiples problemas en la literatura (Baker[16] (1974), Coffman[17] (1976) entre otros).

El problema general de taller o *Shop scheduling* se define de la siguiente manera[18]: Se tienen *n* trabajos y *m* máquinas. Cada trabajo se compone de un conjunto de operaciones O_{ij} con tiempos de procesamiento p_{ij}, y cada operación debe ser procesada en una de las máquinas disponibles. Todos los trabajos tienen relaciones de precedencia entre operaciones; y el objetivo es encontrar una programación factible en la que se optimice una función objetivo.

Según Conway *et al.*[19] (1967), un problema de *Shop Scheduling* puede ser representado de la siguiente manera:

$$\boldsymbol{n/m/C/D}$$

Donde *n* hace referencia al número de trabajos que componen el problema, *m* indica el número de máquinas con las que se dispone, *C* señala el tipo de flujo (siendo F si es *Flow shop*, P si es *Permutation flow shop* y G para *Job shop*), y *D* corresponde a la medida de eficiencia elegida[20].

Existen distintos tipos de clasificación del problema de *Shop scheduling:* de acuerdo al instante en el que llegan los trabajos al proceso (se habla de un problema estático

[16] BAKER, Kenneth R.; BAKER, Kenneth Robert. Introduction to sequencing and scheduling. New York: Wiley, 1974.

[17] COFFMAN, Edward Grady; BRUNO, John L. Computer and job-shop scheduling theory. John Wiley & Sons, 1976.

[18] Ibíd.

[19] CONWAY, Richard W.; MAXWELL, William L.; MILLER, Louis W. Theory of scheduling. Courier Dover Publications, 1967, 294p.

[20] Ibid.

si todos los trabajos llegan al mismo tiempo, o dinámicos si lo hacen de forma gradual o intermitente), pero principalmente, la clasificación del *Shop scheduling* se hace de acuerdo al flujo productivo de los productos, siendo este esquema el más utilizado[21]: *Flow Shop Scheduling Problem* (FSSP), *Job Shop Scheduling Problem* (JSSP) y *Open Shop Scheduling Problem* (OSSP).

4.2.1. Open Shop Scheduling Problem

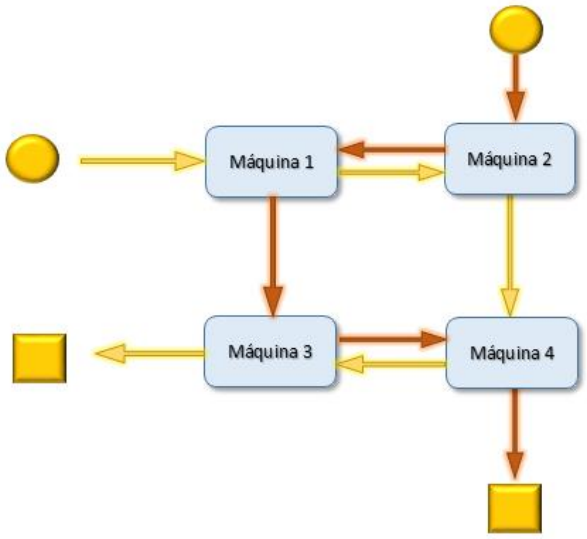
Es un caso especial del *Shop Scheduling*, donde en este problema cada trabajo *i* consiste en un número de operaciones *Oij* que deben ser procesadas en un conjunto de máquinas, donde no se tiene relación de precedencias entre operaciones. En este caso, el problema se centra en encontrar un orden del trabajo (orden de operaciones pertenecientes a un mismo trabajo) y un orden de máquinas (ordenar la operaciones a ser procesadas en la misma máquina). Por lo tanto, las restricciones de un problema de programación *Open Shop* son[22]:

1. *Cada máquina debe procesar una operación a la vez.*
2. *Cada trabajo debe ser procesado por una máquina a la vez.*
3. *Cada operación* $O_{i,j}$ debe ser procesada en $p_{i,j}$ unidades de tiempo por la máquina M_j

[21] LAWLER, Eugene L., et al. Sequencing and scheduling: Algorithms and complexity. En: Handbooks in operations research and management science, 1993, vol. 4, p. 445-522.

[22] LEUNG, Joseph YT. Handbook of scheduling: algorithms, models, and performance analysis. CRC Press, 2004.

Figura 2. Ejemplo Open Shop scheduling problem.

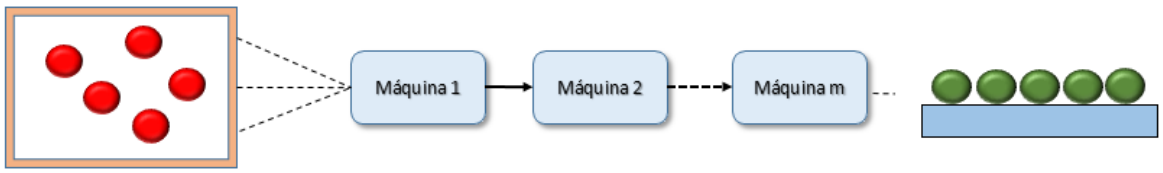


Fuente. Adaptado de MONTREUIL, B. Fractal layout organization for job shop. En: International Journal of Production Research. 1999, vol. 37, no. 3, 501-521

4.2.2. Flow Shop Scheduling Problem

Este problema se caracteriza por tener trabajos con O_{ij} operaciones, las cuales poseen restricción de precedencia de la forma $O_{ij} \rightarrow O_{ij+1}\ \forall \boldsymbol{i} = \mathbf{1}, \mathbf{2}..\boldsymbol{n}$. Es decir, cada trabajo debe pasar en un mismo orden por las máquinas, y todos siguen la misma secuencia de flujo. En este caso, el problema es encontrar un orden de ejecución de cada trabajo para cada máquina.

Figura 3. Ejemplo Flow Shop scheduling problem.



Fuente. Adaptado de MONTREUIL, B. Fractal layout organization for job shop. En: International Journal of Production Research. 1999, vol. 37, no. 3, 501-521.

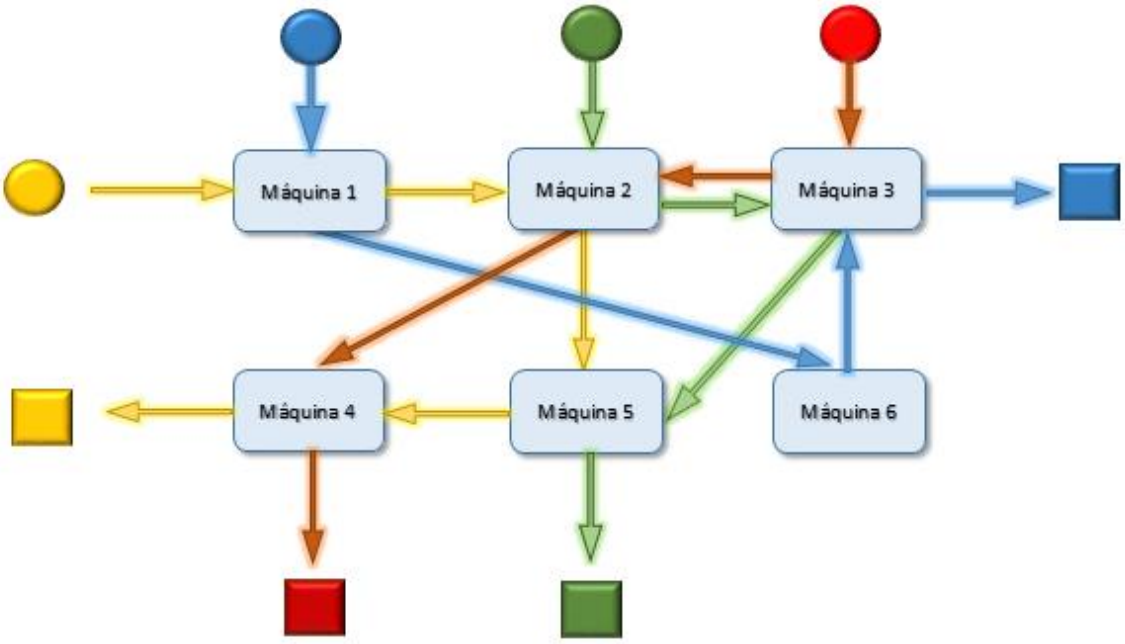
La principal diferencia entre el FSSP y el OSSP es que en el primero las operaciones de cada trabajo deben ser procesadas en orden, como se mencionó anteriormente. En el OSSP el orden en el cual las operaciones son realizadas es irrelevante.

4.2.3. Job Shop Scheduling Problem

Este problema es el caso generalizado del *Flow Shop*, donde se tienen *n* trabajos y *m* máquinas, y cada trabajo presenta una secuencia de operaciones *Oij* distinta. Cada operación debe ser procesada en determinado orden. La relación de precedencia se describe de la siguiente manera:

$$O_{ij} \rightarrow O_{ij+1} \ (j = 1,2, \dots, n_i)$$

Figura 4. Ejemplo Job Shop scheduling problema.



Fuente. Adaptado de MONTREUIL, B. Fractal layout organization for job shop. En: International Journal of Production Research. 1999, vol. 37, no. 3, 501-521.

Los problemas de *scheduling*, se evalúan a través de criterios de optimización (medidas de desempeño) los cuales pueden ser una función arbitraria con valores reales y que ha sido definida en un conjunto de *schedules* posibles para el problema. A continuación se presentan las funciones con las que se puede evaluar el problema del JSSP.

Sabiendo que C_1, C_2,…, C_n son el tiempo de terminación de los trabajos en un *Schedule*, las medidas de desempeño son:

Tiempo máximo de terminación (C_{max})

$$C_{max}=max\{C_i\}$$

Retraso máximo ($L_{máx}$)

$$L_{max} = \max_{1\leq i\leq n}\{L_i\} = \max_{1\leq i\leq n}\{C_i - d_i\}$$

Tardanza máxima (T_{max})

$$T_{max} = \max_{1\leq i\leq n}\{T_i\} = \max_{1\leq i\leq n}\{\max\{0, \quad L_i\}\}$$

El costo máximo (f_{max})

$$f_{max} = \max_{1\leq i\leq n}\{f_i(C_i)\} \; donde \; f_1, f_2, \ldots, f_n \; indican \; las \; funciones \; de \; costos$$

Tiempo total de terminación ($\sum C_i$)

$$\sum C_i = \sum_{i=1}^{n} C_i$$

Tiempo total de terminación ponderado ($\sum w_i C_i$)

$$\sum w_i C_i = \sum_{i=1}^{n} w_i C_i$$

Carga total de la máquina ($\sum C_{max}^{(k)}$)

$$\sum C_{max}^{(k)} = \sum_{i=1}^{n} C_{max}^{(k)}$$

Número de trabajos tardíos ($\sum U_i$)

$$\sum U_i = \sum_{i=1}^{n} U_i$$

Costo total ($\sum f_i$)[23]

$$\sum f_i = \sum_{i=1}^{n} f_i(C_i)$$

De los más comunes según Brucker[24] están el Tiempo máximo de terminación o (*makespan*), Tiempo de flujo de los trabajos *(total flow time*), Tardanza máxima (*earliness*), y Retraso máximo (*tardiness)*, mientras satisface las siguientes restricciones y suposiciones:

- Cada máquina puede procesar un solo trabajo a la vez.
- Cada trabajo sólo puede ser procesado por una máquina al tiempo
- La secuencia de los trabajos a las máquinas está definida
- Todos los trabajos deben ser procesados por cada máquina solo una vez.
- Las máquinas siempre están disponibles y nunca se interrumpen.
- El tiempo de procesamiento de todas las operaciones es conocido.

[23] ZHAO, Chuanli; TANG, Hengyong. Single Machine Scheduling with Nonlinear Processing Times and Learning Effect*. En Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on. IEEE, 2006. p. 7254-7257

[24] BRUCKER, P. Scheduling algorithms. Berlin: Springer, 2007.

El tiempo máximo de terminación o *makespan* corresponde al tiempo total entre el comienzo y la finalización del Schedule determinado.

Para solucionar este tipo de problemas, básicamente se emplean tres clases de técnicas[25]:

4.2.3.1 Reglas de prioridad[26]

Se ordenan los trabajos de acuerdo con un criterio o regla de secuenciación para la construcción de la solución.

Algunas de las reglas más comúnmente usadas en la práctica son:

- SPT (*Shortest Processing Time*): Seleccionar la operación con el tiempo de procesamiento más corto.
- LPT (*Longest Processing Time*): Seleccionar una operación con el tiempo de procesamiento más largo.
- MWR (*Most Work Remaining*): Seleccionar una operación para el trabajo con el mayor tiempo de procesamiento pendiente.
- LWR *(Least Work Remaining*): Seleccionar una operación para el trabajo con el menor tiempo de procesamiento pendiente.
- MOR (*Most Operations Remaining*): Seleccionar una operación para el trabajo con el mayor número de operaciones pendientes.
- LOR (*Least Operations Remaining*): Seleccionar una operación para el trabajo con el menor número de operaciones pendientes.

[25] CASTRILLON, Omar; SARACHE, William; GIRALDO, Jaime. Job Shop problem solution with an intelligent agent. En: Ingeniería y Ciencia, 2009, vol. 5, no 10, p. 75-92

[26] DOMÍNGUEZ, J. A. Dirección de operaciones. Aspectos estratégicos en la producción y los servicios. Ed. Editorial McGraw-Hill. Madrid. 1995.

- EDD (*Earliest Due Date*): Seleccionar un trabajo con la fecha de entrega más temprana.

4.2.3.2 Algoritmos analíticos

Teóricamente pueden garantizar la obtención de soluciones óptimas. Con problemas reales tienen el grave inconveniente de elevar el tiempo de ejecución pues estas técnicas son estáticas y no siempre las condiciones del problema conservan estas características a través del tiempo, sobre todo cuando el número de trabajos *J* y máquinas *M*, aumenta considerablemente.

4.2.3.3 Algoritmos evolutivos

Como se mencionó en el punto anterior, cuando se modelan problemas de la vida real, uno de los problemas más frecuentes es la alta dimensionalidad de espacio de búsqueda. Por medio de la inteligencia artificial, evolución biológica y mecanismos estadísticos, se logran soluciones satisfactorias en un tiempo considerable. No siempre garantizan una solución óptima, pues depende de la complejidad computacional del problema.[27]

Dentro de los problemas anteriormente nombrados, existe una subclase llamada sistema flexible, el cual presenta como característica adicional que cada operación puede ser ejecutada por un conjunto de máquinas (centros de trabajo), capaces de procesar las distintas operaciones.[28]

[27] CASTRILLON, Omar; SARACHE, William; GIRALDO, Jaime. Job Shop problem solution with an intelligent agent. En: Ingeniería y Ciencia, 2009, vol. 5, no 10, p. 75-92

[28] OSORIO, Juan Carlos; MOTOA, Tulio Gerardo. Planificación jerárquica de la producción en un job shop flexible. En: Revista Facultad de Ingeniería, 2014, no 44, p. 160.

4.3. FLEXIBLE JOB SHOP SCHEDULING

Los problemas de secuenciación presentan una serie de variantes dependiendo de la naturaleza y el comportamiento tanto de las operaciones como de las máquinas. En el FJSSP las variantes son, una de las más difíciles de plantear, debido a su alta complejidad computaciones, es aquella en donde las tareas que conforman un trabajo son dependientes y las máquinas son diferentes.[29]

El FJSSP es un problema en el que se deben procesar un conjunto de trabajos o trabajos independientes ($J_1, J_2, J_3, \ldots, J_n$) en M máquinas. Planteado de forma jerárquica o por niveles, se tiene en el nivel superior un conjunto L de centros de trabajo, cada uno conformado por $\boldsymbol{m}$ máquinas, siendo que m ∈ M. Cada centro de trabajo, tendrá asignado un conjunto de trabajos, siendo que cada trabajo tiene asociado un conjunto O_i de operaciones, las cuales deben ser procesadas en determinado orden.

En este tipo de configuración, se pueden presentar dos tipos:

4.3.1. Flexibilidad total

Define que todas las operaciones se pueden ejecutar en todas las maquinas disponibles.

Figura 5. Representación Flexibilidad Total de un Flexible Job Shop Scheduling.

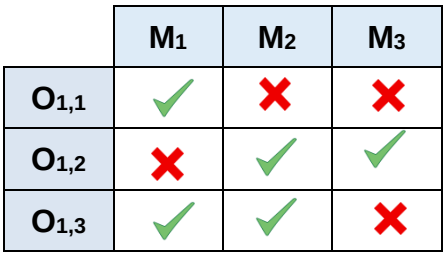
	M_1	M_2	M_3
$O_{1,1}$	✓	✓	✓
$O_{1,2}$	✓	✓	✓
$O_{1,3}$	✓	✓	✓

[29] CORREA, A.; VELASQUEZ, E; LONDOÑO, M. Secuenciación de operaciones para configuraciones de planta tipo job shop flexible: Estado del arte. En: Avances en sistemas e informática, 2008, ed. 5, 151-161.

4.3.2. Flexibilidad parcial

Las operaciones se pueden realizar solo en algunas de las máquinas.

Figura 6. Representación Flexibilidad Parcial de un Flexible Job Shop Scheduling.



La solución, como se expuso a lo largo del documento, consiste en especificar dónde se procesan las operaciones y la secuencia de los mismos en cada una de las máquinas de los centros de trabajo.

4.3.3. Consideraciones

Dentro de las principales consideraciones que se tiene en el *Job Shop* flexible, se encuentran las siguientes[30]:

- Los tiempos de operación son conocidos con certidumbre.
- Todos los trabajos están disponibles para ser procesados en el instante t=0.
- No existen relaciones de precedencia entre trabajos.
- Todas las maquinas se encuentran listas para operar en t=0.
- No se permite la interrupción de los trabajos en las máquinas.
- Todos los trabajos tienen la misma prioridad dentro del sistema.
- No se consideran tiempos de alistamiento.

[30] SIPPER, Daniel. Production: planning, control, and integration/Daniel Sipper, Robert L. Bulfin, Jr. Planeación y control de la producción. 1998.

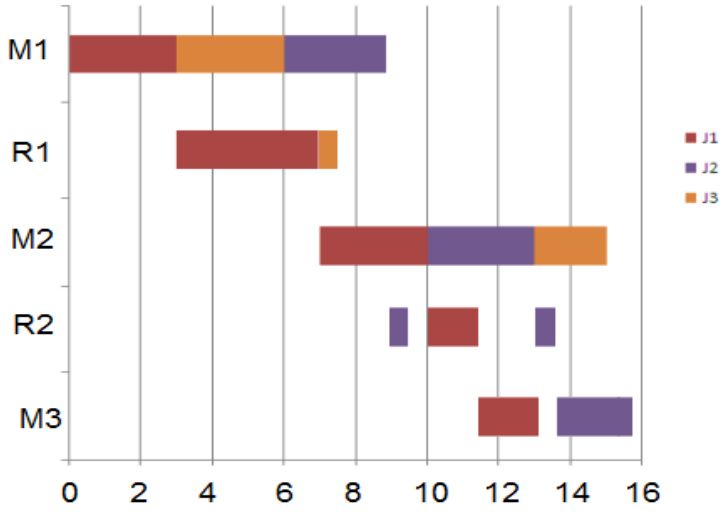
- Se permite la recirculación dentro de la programación, es decir, un trabajo puede visitar 2 veces una misma máquina.
- Restricción de precedencia entre las operaciones de un trabajo.
- Restricción de interferencia para una misma máquina en cualquier instante de tiempo.
- Se considera un trabajo terminado cuando finalicen sus operaciones.

4.3.4. Representación de la solución

4.3.4.1 Diagrama de Gantt

Es la forma gráfica usualmente utilizada para representar problemas de *Shop Scheduling*, donde se especifica en que instante de tiempo se inicia una operación y en cuál de las maquinas se realiza. En el eje vertical se ubican todos los recursos disponibles (incluyendo los de transporte), y en horizontal es una línea de tiempo.

Figura 7. Ejemplo Diagrama de Gantt.



Fuente. Adaptado de SARMIENTO, Cindy Johanna. Metodo particle swarm optimization (PSO- Enjambre de partículas) aplicado al problema de múltiples objetivos del Job Shop Scheduling (JSP) o secuenciamiento de máquinas. Bucaramanga, 2012, 134p.

4.3.4.2 Grafo Disyuntivo

Forma gráfica de representar una solución a un problema, propuesto por Roy y Sussmann, y definido de la siguiente manera:

$$\boldsymbol{G = (N, A, E)}$$

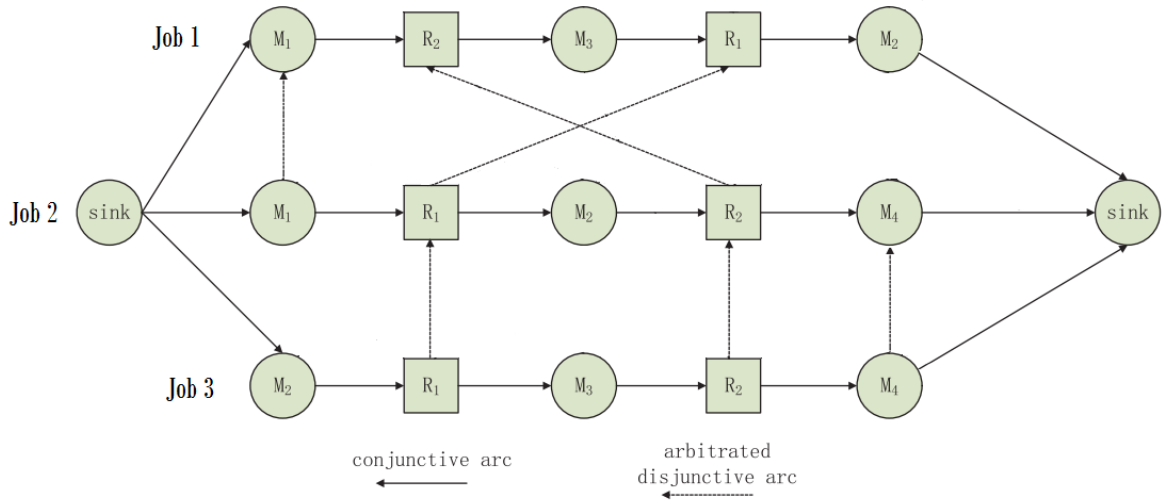
Donde N es el número de nodos o vértices que representan las operaciones, A es el conjunto de arcos dirigidos que hacen referencia a la relación de precedencia entre las operaciones de un mismo trabajo y E contienen los arcos disyuntivos no dirigidos que describen la relación entre las operaciones a ser procesadas por una misma máquina.

Gráficamente, se puede incluir el transporte dentro del grafo de la siguiente manera[31] (teniendo como ejemplo un problema con 3 trabajos, 4 máquinas y 2 recursos de transportes).

$$\boldsymbol{J1}: \boldsymbol{OP_{11}(M_1, M_2) \rightarrow TP_{11} \rightarrow OP_{12}(M_3, M_4) \rightarrow TP_{12} \rightarrow OP_{13}(M_2, M_3)},$$
$$\boldsymbol{J2}: \boldsymbol{OP_{21}(M_1, M_2) \rightarrow TP_{21} \rightarrow OP_{22}(M_2, M_3) \rightarrow TP_{22} \rightarrow OP_{23}(M_3, M_4)},$$
$$\boldsymbol{J3}: \boldsymbol{OP_{31}(M_2, M_4) \rightarrow TP_{31} \rightarrow OP_{32}(M_1, M_3) \rightarrow TP_{32} \rightarrow OP_{33}(M_1, M_4)},$$

[31] ZHANG. Op.cit.

Figura 8. Ejemplo Grafo disyuntivo.



Fuente. Adaptado de ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. En: Computers & Operations Research, 2012, vol. 39, no 7, p. 1713-1723.

En la figura, se observa cuál de las maquinas se asigna a cada operación, y como se establecen las precedencias dentro de las mismas (arco disyuntivo). Las operaciones de transportes son representadas como un nodo de forma cuadrada, conteniendo el recurso que lo realiza.

4.4. MÉTODOS DE SOLUCIÓN PARA EL FJSSP

Para lograr la optimización de un problema se deben aplicar algoritmos en busca de buenos resultados. Con el paso del tiempo, el estudio ha ido evolucionando pasando de métodos exactos hacia heurísticos y finalmente a metaheurísticos. Son estos los tipos de métodos que se presentarán a continuación, definiendo de manera

específica los métodos respectivos de cada tipo más importantes para la búsqueda de solución del problema de interés, el *Job Shop* flexible.[32]

4.4.1. Métodos exactos

Los algoritmos exactos son aquellos que resuelven problemas que pertenecen a la clase P de forma óptima y en tiempo razonable. Algunos ejemplos de estos métodos son los algoritmos voraces, algoritmos de divide y vencerás, algoritmos de ramificación y poda, *backtracking*, etc.[33]

Cuando se habla del *Job Shop*, el algoritmo exacto tradicional es el *Branch and bound* o de ramificación y poda[34]. En él se explora un "árbol" el cual representa el espacio completo de soluciones de un problema y por medio de funciones de acotamiento se descartan las partes del espacio de búsqueda que no pueden contener la mejor solución. Está técnica es bastante eficaz pero como se mencionaba anteriormente, su tiempo de complejidad regularmente es muy alto e inaceptable para problemas *NP Hard.*[35]

El FJSSP es NP-*Hard* por lo que los algoritmos exactos desarrollados no logran soluciones de alta calidad para problemas de gran tamaño en tiempo razonable y debido a esto diferentes investigadores se han enfocado en heurísticas y meta-heurísticas.

---

[32] VIDAL. Op. cit.

[33] DUARTE, Abraham; PANTRIGO, Juan; GALLEGO Micael. Introducción a la optimización. En: Metaheuristicas. Madrid: Dykinson, 2008. p. 1-14.

[34] SAIDI-MEHRABADA, Mohammad y FATTAHI, Parviz. Flexible job shop scheduling with tabu search algorithms. En: The Internal Journal Of Advanced Manufacturing Technology. Mayo, 2006. vol. 32. p. 563-570.

[35] VIDAL. Op. cit.

4.4.2. Heurísticas

Como respuesta a las falencias en cuanto al tiempo, costo y menor flexibilidad que tienen los métodos exactos con respecto a la solución problemas tales como los de clase *NP* existen los algoritmos heurísticos[36]. Adicionalmente, el método heurístico se puede utilizar como parte de un procedimiento global en el cual proporciona una buena solución inicial de partida o participa en un paso intermedio del mismo. Según Fernandez *et al*. (1996), se puede definir a un algoritmo heurístico como: "Procedimiento para resolver un problema de optimización bien definido mediante una aproximación intuitiva, en la que la estructura del problema se utiliza de forma inteligente para obtener una buena solución"[37].

Los algoritmos heurísticos se caracterizan por ser completamente dependientes del problema (o familia de problemas) que resuelve, de forma que al cambiar el problema se debe diseñar un nuevo algoritmo exacto y demostrar su optimalidad.

Los métodos heurísticos se pueden dividir en métodos constructivos y métodos de búsqueda local.

***Métodos constructivos**:* Son procedimientos iterativos capaces de construir la solución. Usualmente son métodos deterministas y suelen estar basados en la mejor elección de cada iteración de manera que paso a paso se llega a la solución del problema.

[36] KOKASH, Natalia. An introduction to heuristic algorithms. Departamento de Informática y Telecomunicaciones. 2005.

[37] FERNÁNDEZ, Adenso D., et al. Optimización heurística y redes neuronales. ed. Paraninfo SA, Madrid, España, 1996.

Para los problemas de *Scheduling*, las heurísticas más usadas son el algoritmo de Giffler & Thompson (implementación reglas de despacho) y el algoritmo de cuello de botella móvil.

Métodos de búsqueda local*:* Se define de la siguiente manera: cada solución *x* tiene un conjunto de soluciones asociadas *N(x)*, que se denomina entorno de *x*. Cada solución *x'* de su entorno *N(x)* puede obtenerse directamente a partir de *x* mediante una operación llamada movimiento.

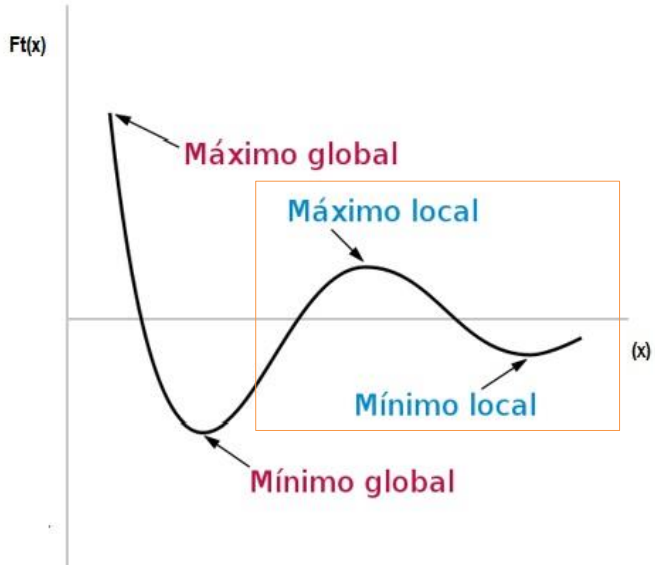
A diferencia del anterior, el método de búsqueda local es un procedimiento que parte de una solución factible dada y a partir de ella intenta mejorarla realizando en cada paso un movimiento de una solución a otra con mejor valor.

Existen diferentes estrategias las cuales se puede aplicar para la utilización de este método.

- Estrategia de búsqueda local 1 (*first improvement*): partiendo de una solución factible como su definición lo indica, esta estrategia examina su vecindad y se selecciona el primer movimiento que produzca una mejora a la solución actual para así mejorar progresivamente.
- Estrategia de búsqueda local 2 (*best improvement*): partiendo de una solución factible como su definición lo indica, esta estrategia examina su vecindad y se selecciona el mejor movimiento que produzca, en el caso de maximización, el mayor incremento en la función objetivo.
- Estrategia aleatorizada: para una solución factible dada y una vecindad asociada a la solución, se selecciona aleatoriamente soluciones vecinas de esa vecindad.

El principal problema que presentan los algoritmos heurísticos es la incapacidad de salir de óptimos locales tal como se muestra en la figura 9 (ejemplo basado en el método de búsqueda local).[38]

Figura 9. Ejemplo de la incapacidad de los algoritmos heurísticos.



Fuente. Adaptado de DUARTE, Abraham; PANTRIGO, Juan; GALLEGO Micael. Introducción a la optimización. En: Metaheuristicas. Madrid: Dykinson, 2008. p. 1-14.

Adicional a esto, la falta de flexibilidad cuando la formulación del problema cambia y el no tener la capacidad de generar soluciones de alta calidad, muy a menudo incentiva a investigadores en la producción de nuevos métodos en los cuales se introducen otros algoritmos de búsqueda más inteligentes de nivel superior, denominados metaheurísticas.[39]

[38] DUARTE. Op. cit.

[39] VIDAL. Op. cit.

4.4.3. Metaheurísticas

Los algoritmos metaheurísticos son procedimientos iterativos que guían una heurística combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

Los profesores Osman y Kelly en 1995 introducen la siguiente definición: "Los procedimientos metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos"[40]. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

La utilización de las metaheurísticas se recomienda cuando:

- No existe un método exacto de resolución o éste requiere mucho tiempo de cálculo o memoria.
- No se necesita una solución óptima, sino una solución satisfactoria que oriente sobre el comportamiento del problema.
- Los datos son poco fiables.
- Las limitaciones de espacio, tiempo, etc., obliguen al empleo de métodos de respuesta rápida.
- Como paso intermedio den la aplicación de otro algoritmo.

[40] OSMAN, Ibrahim H.; KELLY, James P. Meta-heuristics: an overview. En *Meta-Heuristics*. Springer US, 1996. p. 1-21.

A continuación se presentan los siguientes algoritmos metaheurísticos para la solución de problemas de optimización en especial del *Job shop* flexible.[41]

Búsqueda Tabú

La búsqueda tabú (TS, por sus siglas en inglés) es un método metaheurístico propuesto por Glover en la época de los 80.[42]

El algoritmo TS se considera una técnica de búsqueda local puesto que realiza una exploración a través de toda la vecindad estudiando adecuadamente los óptimos locales.

Esta última característica se cumple por media del uso de estructuras de memoria aumentando así el rendimiento del mismo. Según Nowicki & Smutnicki (1996) la aplicación de un algoritmo TS en un ambiente *Job Shop* se presenta de la siguiente manera: inicialmente se define un movimiento, acción mencionada en la definición de la búsqueda local, la cual se define como una perturbación particular que transforma una solución en otra. Partiendo de una solución o secuenciación inicial, el algoritmo lo que hace es buscar en el entorno de dicha solución, una nueva secuencia para una próxima iteración. De esta manera se genera un camino o recorrido por el conjunto de secuencias. Con el fin de evitar que el camino se convierta en un ciclo en el algoritmo, se guardan las últimas k secuencias generadas las cuales se denominan lista Tabú. Al concluir un camino o trayecto, se puede aplicar una estrategia de diversificación en la cual se pretende retroceder el proceso

[41] FATTAHI, Parviz; MEHRABAD, Mohammad Saidi; JOLAI, Fariborz. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. En: Journal of Intelligent Manufacturing, 2007, vol. 18, no 3, p. 331-342.

[42] SARIÇIÇEK, İnci y CENK, Çelik. Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness. En: Applied Mathematical Modelling. 2011. vol. 35. p. 4117-4126.

de manera que se aplique el TS partiendo de una nueva solución generando así un nuevo camino. Finalmente, la mejor solución obtenida resulta de la mejor secuencia entre las examinadas anteriormente.

El propósito de la lista Tabú es prevenir la búsqueda de su degeneración al entrar en un ciclo entre mismas soluciones. Normalmente, esto se realiza conformando la lista tabú teniendo en cuenta alguna caracterización de las últimas s movidas realizadas. Dauzère-Pérès y Paulli[43] (1997) probaron tres tipos diferentes de listas tabú como se presentan a continuación.

Sabiendo que *i* es la operación movida entre las operaciones *j* y *k, s* y *t* son las operaciones antes y después de *i* en la secuencia actual y *a(i)* es la máquina de la operación i asignada antes del movimiento.

- (*s,t*) es incluida a la lista tabú. Un movimiento (*i',j',k'*) es prohibido sí (*j',k'*) pertenece a la lista tabú.
- (*i, a(i)*) es incluida a la lista tabú. Un movimiento (*i',j',k'*) es prohibido sí (*i', a(j')*) (*o (i', a(k')) si j'=0*) pertenece a la lista tabú.
- (*i,j*) es incluida a la lista tabú. Un movimiento (i',j',k') es prohibido sí (*i', j'*) o (*k', i'*) pertenece a la lista tabú.

Según el artículo elaborado por estos autores, se expone que la tercera lista es aquella que presenta los mejores resultados y por tanto es la más recomendada. Debido a esto, las principales decisiones a tener en cuenta son: especificación de la estructura de la vecindad, la característica de movimiento, la longitud de la lista tabú, el criterio de aspiración (para determinar cuándo una restricción tabú puede ser anulada) y el criterio de parada.

[43] DAUZÈRE-PÉRÈS, Stéphane; PAULLI, Jan. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Annals of Operations Research, 1997, vol. 70, p. 281-306.

En conclusión, la búsqueda Tabú consiste en un procedimiento que restringe la búsqueda y evita los mínimos locales, almacenando la historia de búsqueda por medio del uso de estructuras mentales. De igual manera prohíbe movimientos entre vecinos que cumplan ciertas propiedades y así no duplicar soluciones anteriores.

Recocido Simulado

En 1983, Kirpatrick et al.[44] proponen este procedimiento basado en la analogía con el comportamiento de un sistema físico al someterlo en un baño de agua caliente. Su nombre se justifica por el templado posterior o enfriado con el que se producen ciertas sustancias tales como la cristalización del vidrio o el recocido del acero. El recocido simulado o simulated annealing (SA) establece un paralelismo entre el proceso de las moléculas de una sustancia en diferentes niveles energéticos buscando un equilibrio y las soluciones visitadas por un procedimiento de búsqueda local. El espacio de configuraciones no vendrá ya dado por las posiciones de las moléculas, sino por los valores de una variable de interés, que en el caso del FJSSP será la secuencia de tareas que se desean procesar, y el papel de la energía lo asumirá la función que se intenta minimizar (costo o tiempo por ejemplo). Siendo así, el recocido simulado es un procedimiento basado en búsqueda local en el cual se usan técnicas de optimización no determinista, es decir, no buscan la mejor solución del entorno de la solución actual sino que generan aleatoriamente una solución cercana y la aceptan como la mejor si tiene menor costo, o aceptan soluciones de no mejora de acuerdo a unas probabilidades. Estas probabilidades están representadas por $e^{-\frac{\Delta}{T}}$, usando la temperatura T del sistema como parámetro de control[45]. La temperatura será gradualmente reducida en cada iteración y de esta manera las probabilidades de tener soluciones de no mejora son cada vez más

[44] KIRKPATRICK, Scott; GELATT, Daniel y VECCHI, Mario P. Optimization by simulated annealing. En: Science, 1983. p. 671-680.

[45] FATTAHI. Op. cit.

pequeñas conforme avanza el procedimiento y nos acercamos a la solución óptima. El recocido simulado se caracteriza por tener la habilidad de salir de óptimos locales al aceptar movimientos de no mejora o malos movimientos en los estados intermedios, siendo estos tan poco probables al final, que al no encontrar movimientos de mejora, el algoritmo finaliza.[46]

Algoritmos genéticos

Los algoritmos genéticos fueron introducidos por Holland[47] (1975) inspirándose en el proceso observado en la evolución natural de los seres vivos y ha sido aplicado en diferentes campos tales como la matemática, la ingeniería, la biología y las ciencias sociales[48]. A lo largo de las generaciones, las poblaciones evolucionan en la naturaleza acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin (1859). Así mismo, los AG comienzan con un conjunto inicial de soluciones, llamadas poblaciones. Cada solución se considera un cromosoma como punto de búsqueda en el espacio, el cual está compuesto por genes[49]. El AG funciona de manera iterativa y cada iteración es llamada una generación. En las diferentes generaciones, se combinan los genes de los progenitores por medio de dos operadores genéticos, cruce y mutación. A medida en que las iteraciones aumentan, se supone que los cromosomas tendrán los mejores genes de sus padres, por lo tanto, la calidad promedio de las soluciones

[46] MARTÍ, Rafael. Procedimientos Metaheurísticos en optimización combinatoria. [En línea] Universidad de Valencia, Departamento de Estadística e investigación operativa. (2003). P. 1-60. [Consultado 28 ene. 2013]. Disponible en: <http://www.uv.es/~rmartí/paper/cpapers.html>

[47] HOLLAND. Op. cit.

[48] GOLDBERG, David E. Genetic Algorithms in Search, Optimization and Machine Learning. En: Addison-Weasley Longman Publishing Co. 1989. p. 372. ISBN: 0201157675.

[49] ZANDIEH, M.; MAHDAVI, I y BAGHERI, A. Solving the Flexible job-Shop Scheduling Problem by a Genetic Algorithm. En: Journal of Applied Sciences. 2008. ISBN 1812-5654.

será siempre mejor que la anterior generación. Este proceso de evolución se repite hasta que algún criterio de parada tal como número de iteraciones o la diferencia de avance en la solución menor a una tolerancia establecida es previamente es dado.

Colonia de Hormigas

Los algoritmos ACO (*Ant Colony Optimization*) son modelos inspirados en el comportamiento de colonias de hormigas reales las cuales, a pesar de ser casi ciegas, pueden seguir la ruta más corta en su camino de ida y vuelta entre la colonia y una fuente de abastecimiento. Esto se debe a que las hormigas depositan en su camino una hormona denominada feromona, la cual, de cierta manera "transmite información" haciendo que más hormigas sigan este camino. Cuantas más hormigas recorren una trayectoria, más intenso es el olor del rastro y de igual manera ante la ausencia de hormigas la feromona se evapora y por lo tanto se debilita. Por lo tanto, los algoritmos ACO son procesos iterativos en donde en cada iteración se lanza una colonia de m hormigas y cada una de las hormigas construye una solución al problema de manera probabilística guiándose por un rastro de feromona artificial y por una información calculada a priori de manera heurística. El resultado final es la mejor solución encontrada a lo largo de todas las iteraciones realizadas.[50]

Según Khalouli y Ghedjati[51], para modelar el comportamiento de las hormigas se deben definir los siguientes componentes:

- Representación adecuada de feromona.
- Mecanismo de actualización de la cantidad de feromona.

[50] VIDAL. Op. cit.

[51] KHALOULI, S.; GHEDJATI, F.; HAMZAOUI, A. A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence*, *23*(5), 765-771. 2010.

- Función que pueda brindar información sobre el problema específico.

Optimización por enjambre de partículas (PSO)

La optimización de enjambre de partículas es un método para funciones no lineales en espacios continuos y discretos desarrollado por Kennedy & Eberhart[52] en 1995, basado en la analogía del comportamiento social de animales tales como el vuelo de aves, el movimiento de peces y la teoría de enjambre. Es considerado como una de las técnicas de convergencia más rápidas y un algoritmo muy ajustable a funciones multimodales.

En este sistema, la búsqueda se realiza a partir de una población de partículas que corresponde a los individuos, cada uno de los cuales representa una solución candidata al problema. Estos individuos cambian de estado al "volar" a través del espacio de búsqueda hasta que se ha encontrado un estado relativamente estable.[53]

PSO tiene algunas características atractivas. Al tener memoria, el conocimiento de buenas soluciones es retenido para todas las partículas. Adicionalmente, tiene cooperación constructiva entre partículas, y las partículas en el enjambre comparten información entre ellos. Debido a su simple concepto, fácil implementación, y rápida

[52] MOSLEHI, Ghasem y MEHDI, Mahnam. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. En:International Journal of Production Economics 129.2011. p.14-22.

[53] MUÑOZ, Mario A. Estrategias Evolutivas aplicadas a los Algoritmos de Enjambres para el control de sistemas complejos. Tesis de Maestría. Santiago de Cali: Universidad del Valle. Facultad de Ingeniería y Electrónica. 2008. 90 p.

convergencia, PSO ha sido de centro de atención y gran aplicación en diferentes campos, principalmente para problemas de optimización continua.[54]

Método Híbrido

El descubrimiento y aplicación de los algoritmos metaheurísticos híbridos ha sido de gran interés académico. Al combinar conceptos o componentes de diferentes metaheurísticas se incrementan las fortalezas y eliminan las debilidades de los mismos. Por lo tanto, la efectividad de la búsqueda en el espacio de soluciones es mejorada y nuevas oportunidades aparecen haciendo más poderosos y flexibles los métodos de búsqueda.

Talbi[55] (2002) propuso una clasificación para los algoritmos metaheurísticos híbridos como se presenta a continuación.

1. Intercambio de componentes (métodos basados en población con métodos de búsqueda local)

2. Búsqueda cooperativa (intercambio de información entre dos o más algoritmos metaheurísticos)

3. Algoritmos metaheurísticos integrados y métodos sistemáticos.

[54] LIU, B.; WANG, L.; JIN, Y. H. An effective PSO-based memetic algorithm for flow shop scheduling. Systems, Man, and Cybernetics, Part B: Cybernetics, En: IEEE Transactions, 2007, 37(1), 18-27.

[55] TALBI, E.-G. A taxonomy of hybrid metaheuristics. *Journal of heuristics*, 2002, vol. 8, no 5, p. 541-564.

Gran mayoría de los métodos híbridos para *Shop Scheduling* pertenecen a la primera clasificación mencionada[56]. Zhang[57], presentó un algoritmo híbrido basado en un PSO y un VNS para resolver un FJSSP multiobjetivo.

4.5. DESCRIPCIÓN DEL ALGORITMO GENÉTICO

Los Algoritmos Genéticos (AG) son métodos adaptativos basados en el proceso genético de los organismos vivos. Según los principios postulados por Darwin en 1859 acerca de la selección natural y la supervivencia de los más fuertes, se imita el proceso evolutivo de la naturaleza para crear los algoritmos genéticos capaces de ir creando soluciones para problemas del mundo real.

Los principios básicos fueron establecidos por primera vez por Holland[58] (1975) y a lo largo de los años han sido descritos en diferentes textos tales como los propuestos por Goldberg[59] (1989), Michaelewicz[60] (2013), Reeves[61] (1993), entre otros.

---

[56] TALBI. Op. cit.

[57] ZHANG, G. Hybrid variable neighborhood search for multi objective flexible job shop scheduling problem. In Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on (pp. 725-729).

[58] HOLLAND. Op. cit.

[59] GOLDBERG. Op. cit.

[60] MICHALEWICZ, Zbigniew. *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013.

[61] REEVES, Colin R. Modern heuristic techniques for combinatorial problems. En: John Wiley & Sons Inc., 1993.

Los Algoritmos genéticos (AG) trabajan con una población de individuos donde cada uno de los cuales representa una solución factible a un problema dado. Se puede pensar que cada uno de ellos representa un punto en el espacio de búsqueda como candidatos de solución. Cada individuo o cromosoma está compuesto por información genética. Se le asignará un valor o puntuación relacionado con la bondad de la solución y representa lo que en la naturaleza corresponde al grado de efectividad de un organismo para competir por unos determinados recursos. El AG funciona de manera iterativa y cada iteración es llamada una generación. En las diferentes generaciones, se combinan los genes de los progenitores por medio de operadores genéticos. A medida en que las iteraciones aumentan, se supone que los cromosomas tendrán los mejores genes de sus padres, por lo tanto, la calidad promedio de las soluciones será siempre mejor que la anterior generación. Este proceso de evolución se repite hasta que algún criterio de parada tal como: tamaño de la población, número de iteraciones, probabilidad de cruce y mutación, entre otros, es dado.

Siendo así, el esquema general para la implementación del Algoritmo Genético se puede definir de la siguiente manera[62]:

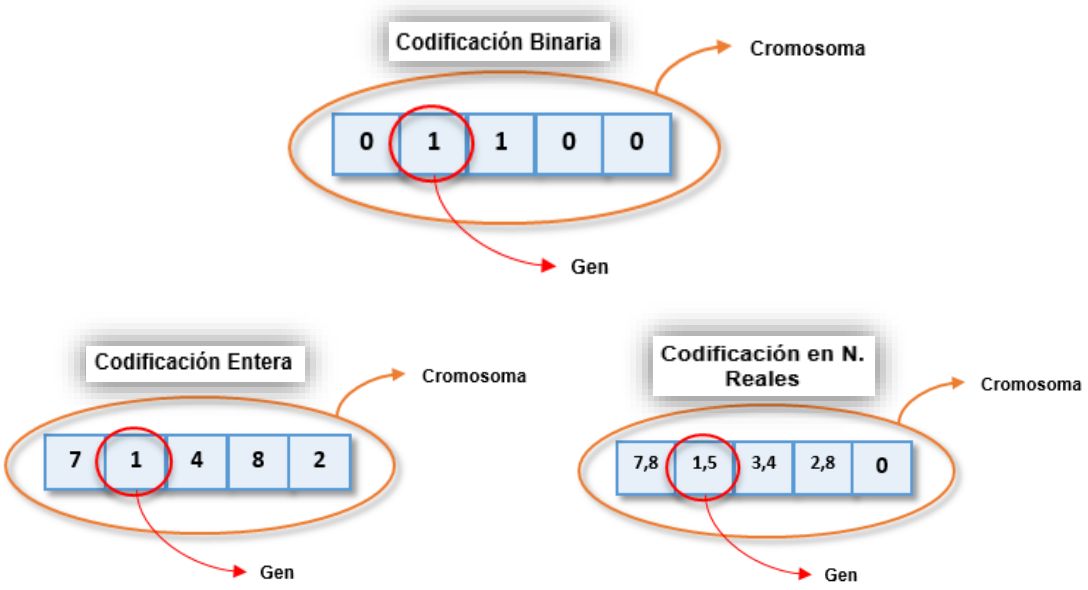
4.5.1. Esquema general

1) Inicialización: generación de una población inicial con *P* soluciones llamadas cromosomas. Los cromosomas son codificaciones de soluciones al problema, los cuales pueden ser representados de forma binaria, entera o en números reales. Generalmente, la población inicial es creada de manera aleatoria[63].

[62] PARJAPATI, S. K.; JAIN, A. Optimization of Flexible Job Shop Scheduling Problem with Sequence Dependent Setup Times Using Genetic Algorithm Approach.

[63] DEB, K. Optimization for Engineering Design, Algorithm and Examples, Prentice Hall, New Delhi, India. 2003.

Figura 10. Tipos de codificación de un cromosoma.



Fuente. Adaptado de LÓPEZ DÍAZ, José Carlos. Un algoritmo genético con codificación real para la evolución de transformaciones lineales. 2010.

Un cromosoma hace referencia a una solución potencial del problema, y el conjunto son denominados población. Cada individuo es evaluado a través de una función que determina su *fitness* (qué tan bueno es el individuo en cuanto al criterio de optimización).

2) Evaluación: con el fin de indicar sí los individuos de una población representan o no buenas soluciones al problema del FJSSP, se define y aplica una función de evaluación o *fitness* la cual se puede considerar como la probabilidad de que un cromosoma sea lo suficientemente apto para reproducirse a partir de los operadores que se presentarán más adelante.

3) Selección: Se escogen individuos o cromosomas entre la población para efectuar la reproducción. Entre más se adapte el individuo al problema (puntuación), existe mayor probabilidad de que el mismo sea seleccionado para reproducirse con otro individuo seleccionado de igual forma, pues el operador de selección genera a partir de la población actual una población intermedia del mismo tamaño, reproduciendo

con un mayor número de copias a los individuos más aptos y eliminando o asignando un menor número de copias a los individuos menos aptos.

A continuación se presentan los operadores de selección mayormente utilizados[64].

- Selección de Ruleta

También conocida como “Selección proporcional a la función del desempeño”, consiste en asignar a cada uno de los cromosomas o individuos de la población un porcentaje de una ruleta, proporcional a su desempeño, de manera que los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores y la suma total de los porcentajes asignados sea igual a 1.

La probabilidad asociada a su selección está dada por:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

Donde N es el número de individuos existentes y *fi* el desempeño del *i-ésimo* individuo.

Para seleccionar un individuo basta con generar un número aleatorio del intervalo [0 1] y devolver el individuo situado en esa posición de la ruleta.[65]

- Selección por torneo

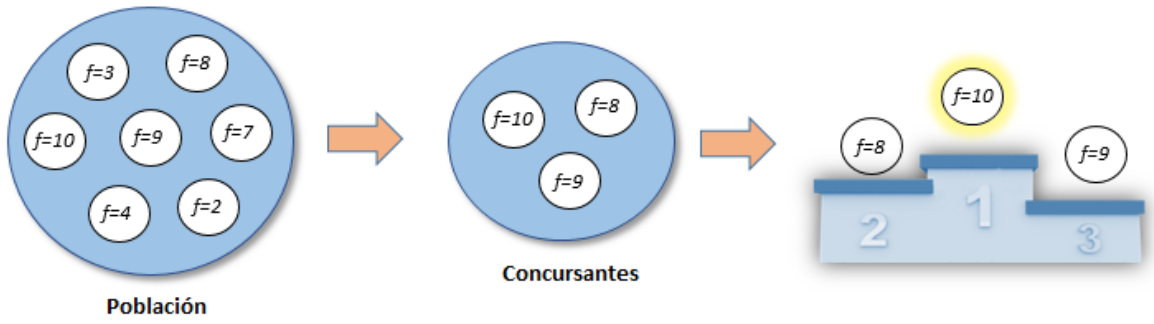
Operador de selección que escoge un subconjunto al azar de la población total con el fin de compararlos y de cierto modo concursar por el primero puesto (mejor desempeño). Entre más grande sea el subconjunto, la presión de selección será

[64]BLICKLE, T.; THIELE, L. A Mathematical Analysis of Tournament Selection. In ICGA (pp. 9-16). Chicago. 1995.

[65] Ibid.

más alta al igual que la probabilidad de reproducir individuos de bajo desempeño. Su principal ventaja es la velocidad de aplicación, ya que no es necesario evaluar ni comparar la totalidad de la población.

Figura 11. Representación gráfica de Selección por Torneo.



- Selección Basada en Ranking

Baker[66] (1985) introdujo la Selección por Ranking al Algoritmo Genético, en donde básicamente, se ordena la población de acuerdo a su *fitness* (más apto a menos apto) asignando el número de copias que cada individuo debe recibir de acuerdo a una función de asignación (ranking).

4) Cruce: Trata de un operador cuya labor es elegir un lugar o punto de cruce aleatorio, y cambiar las secuencias antes y después de esa posición entre dos individuos o cromosomas para crear una nueva descendencia. Imita la recombinación biológica entre dos organismos haploides.

Los operadores de cruce comúnmente utilizados en el Algoritmo Genético son[67]:

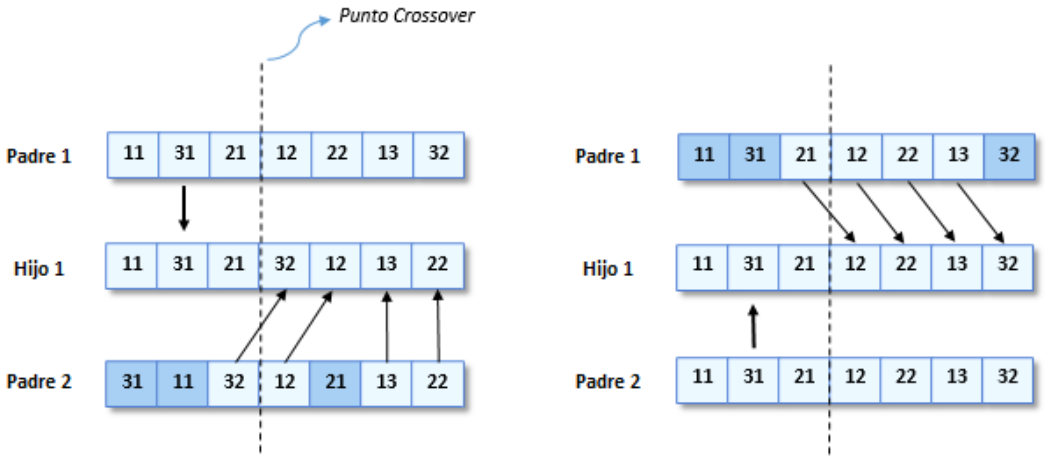
[66]BAKER, James Edward. Adaptive selection methods for genetic algorithms. En Proceedings of an International Conference on Genetic Algorithms and their applications. 1985. p. 101-111.

[67] BLICKLE. Op. Cit.

- Crossover 1 punto

Ilustrado en la figura 12, el crossover de un punto trata de la división e intercambio de genes de los cromosomas padres para crear dos cromosomas hijos con mayor probabilidad de desempeño. El primer cromosoma hijo será producto de la información genética de uno de los padres hasta un punto seleccionado de cruce y el resto se copiará del otro progenitor. Los genes que no fueron copiados en cada uno de los padres darán lugar al segundo cromosoma hijo.

Figura 12. Representación gráfica de Crossover de 1 punto.

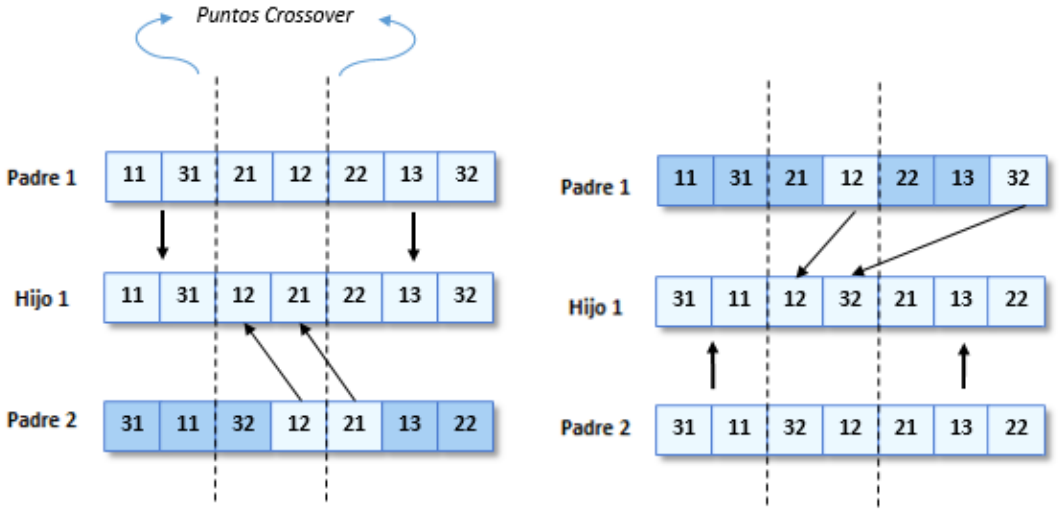


- Crossover 2 puntos

A diferencia del caso anterior, en este tipo de cruce se definen dos puntos de división de los cromosomas padre. El primer cromosoma hijo será producto de la información genética de uno de los padres desde el inicio al primer punto de cruce. Del otro padre se copiarán los genes que siguen desde el primer punto definido a un segundo punto de cruce. Por último se vuelve a tomar la información del primer padre a partir del segundo punto de cruce al final del cromosoma padre. De igual manera, la información que no fue copiada de cada uno de los padres, dará lugar al segundo cromosoma hijo.

Según las investigaciones realizadas por De Jong[68] (1975) acerca del operador de cruce basado en múltiples puntos, concluye que el cruce basado en dos puntos es el que mejores resultados arroja en comparación con 3 o más puntos.

Figura 13. Representación gráfica de Crossover 2 puntos.



- Crossover Uniforme

Este operador, es completamente diferente a los anteriores ya que en él, interviene una máscara de cruce con valores binarios.[69] Los número 1 que se encuentren en la máscara corresponderán a un cromosoma padre y los número 0 al otro respectivamente. De esta manera, con base en la ubicación de cada número en la máscara, se copiarán y ubicarán los genes en el cromosoma hijo resultante. Para producir el segundo cromosoma hijo, se intercambian los papeles de los padres, o

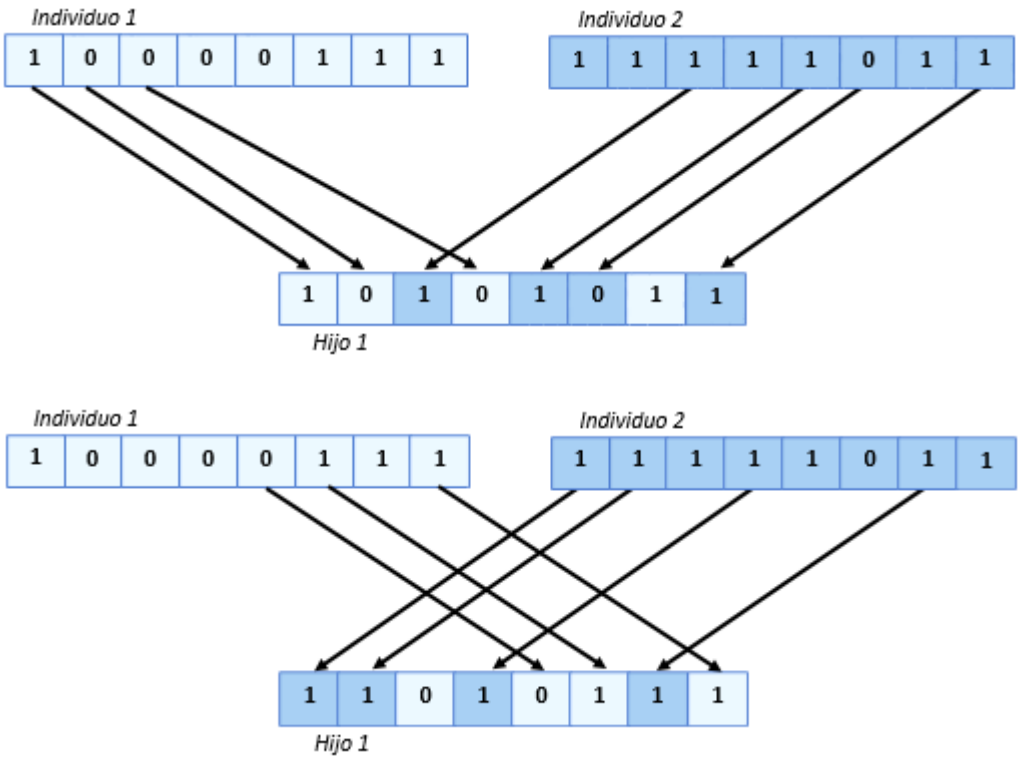
[68] DE JONG, Kenneth Alan. An analysis of the behaviour of a class of genetic adaptive systems. Tesis doctoral, University of Michigan. 1975.

[69] RIVERO, Daniel; RABUÑAL, Juan Ramon; DORADO, Julian; PAZOS, Alejandro. Introducción a los algoritmos genéticos y la programación genética. Universidade da Coruña, Servicio de Publicacións, 2010.

bien la interpretación de los uno y ceros de la máscara de cruce como se muestra en la Figura 14.

La probabilidad de cruce, se define como la relación entre el número de hijos producidos en cada generación y el tamaño de la población, es decir, controla el número esperado de cromosomas que se someten a la operación de cruce[70].

Figura 14. Representación gráfica de Crossover Uniforme.



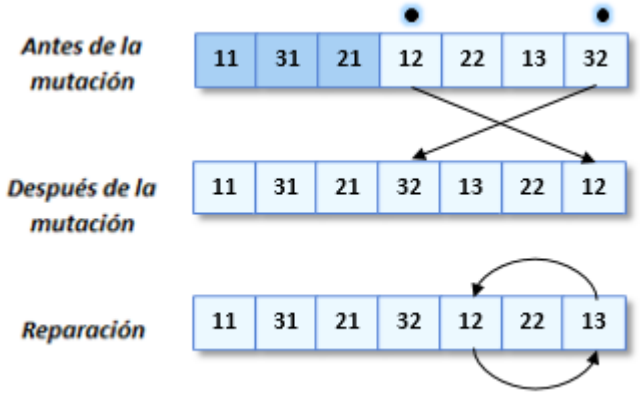
5) Mutación: Este operador produce variaciones aleatorias en un individuo o cromosoma, modificando aleatoriamente la función de los padres y generando así un nuevo individuo. La probabilidad de mutación, se define como el número total de genes en la población que deben ser mutados, es decir, controla el porcentaje en el

[70] HERNÁNDEZ, J. L. Algoritmos genéticos y su aplicación en optimización de redes (Doctoral dissertation, Facultad de Informática). 1998.

cual se introducen nuevos genes en la población. Lo anterior, ayuda a mantener un nivel razonable de diversidad de la población, y proporciona un mecanismo para escapar de óptimos locales[71]. Según T. Abdelmaguid[72], los siguientes operadores de mutación son considerados como adecuados para la codificación del Algoritmo Genético:

- Mutación *Flip*: se selecciona de manera aleatoria dos operaciones para ser intercambiadas una por la otra teniendo en cuenta el inicio de la mutación y el fin de la misma.

Figura 15. Representación gráfica de la Mutación *Flip.*

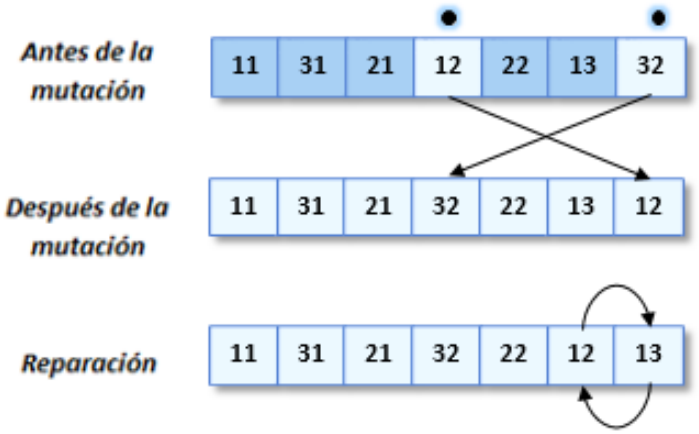


- Mutación *Swap*: se seleccionan de manera aleatoria dos operaciones para ser intercambiadas una por la otra. El intercambio sólo se realizará con operaciones de distintos trabajos. Una vez mutado el cromosoma, se repara sí es necesario con el fin de cumplir con la restricción de precedencia.

[71] PARJAPATI, S. K.; JAIN, A. Op. Cit.

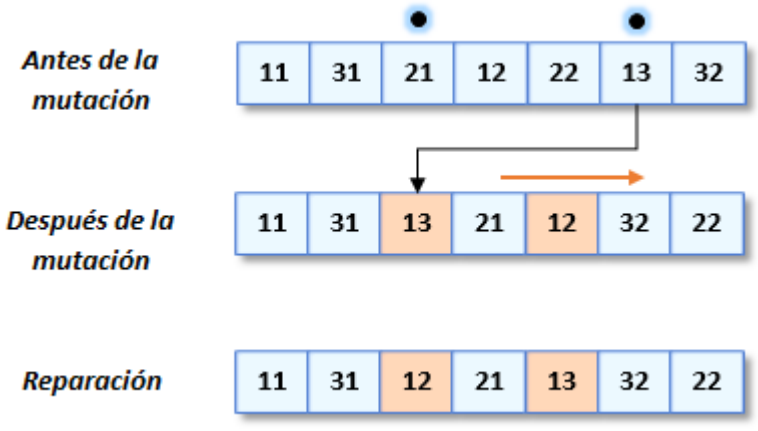
[72] ABDELMAGUID, Tamer F., et al. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. *International journal of production research*, 2004, vol. 42, no 2, p. 267-281.

Figura 16. Representación gráfica de la Mutación Swap.



- Mutación *Slide*: Este tipo de mutación, a diferencia del anterior, selecciona aleatoriamente tanto una operación como una posición para ubicar la misma. Una vez la operación está en su nueva posición, las demás operaciones siguientes se desplazarán una posición. De ser necesario, el cromosoma se reparará con el fin de cumplir con la restricción de precedencia.

Figura 17. Representación gráfica de la Mutación *Slide.*



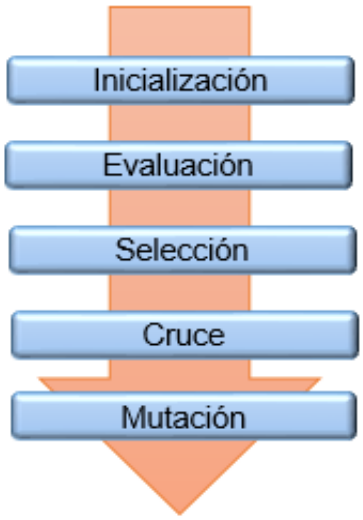
De este modo, los operadores de cruce y mutación pueden verse como formas de mover una población en lo que se conoce como “paisaje potencial” según Sewell

Wright, y un algoritmo genético como un método de búsqueda de ese paisaje para cadenas altamente cualificadas.

Por lo tanto, para poder resolver un problema mediante un algoritmo genético, se deben definir un cierto número de componentes tales como:

- Representación genética de las soluciones potenciales del problema.
- Modo de crear la población inicial.
- Función de evaluación que juega el papel del entorno.
- Operadores genéticos de selección, cruce y mutación que simulan la evolución de las poblaciones.
- Valores de los parámetros o criterios de parada del algoritmo: tamaño de la población, número de iteraciones, probabilidad de cruce y mutación, entre otros.[73]

Figura 18. Representación gráfica del esquema general del Algoritmo Genético.



[73] GONZALEZ, Miguel A. Soluciones Metaheuristicas al "Job-Shop Scheduling Problem with Sequence-Dependent Setup Times". Tesis Doctoral. España: Universidad de Oviedo. Departamento de informática. Julio 2011. 2810 p.

5. ESTADO DEL ARTE

El *Job Shop Scheduling Problem* (JSSP) es un problema clásico de la programación de operaciones, que contiene dentro de sus características unas tareas por cumplir compuestas por operaciones, recursos limitados para el procesamiento de tareas y un conjunto de restricciones; todas estas que acotan una serie de objetivos a optimizar.

Inicialmente, no se confirma con seguridad quien fue el pionero en proponer el JSSP, pero autores como Jain (1999), Graham *et al.* (1979); y Conway *et al.* (1967), declaran a S.M. Johnson[74] (1954) como pionero en el estudio de problemas de *Scheduling* a través de su artículo: "*Optimal two and three stage production schedules with set-up time included*", donde describe un problema sobre *flow shop* resuelto por medio de un algoritmo propuesto y elaborado por dicho autor.

Posteriormente, Blazewicz *et al*. (1996) describen que son Muth & Thompson[75] (1963) quienes tratan el *Job Shop* y problemas de programación, editando un libro llamado "*Industrial scheduling*", que constituye la base para investigaciones posteriores sobre estos temas.

El problema del *Job Shop Scheduling* ha sido una línea de investigación de amplio interés debido a sus beneficios en cuanto a problemas reales, su impacto en el mundo empresarial y su alta complejidad computacional de tipo *NP-Hard* (Garey, 1979), donde se expone que debido a esto, soluciones metaheurísticas son las más acertadas para este tipo de problemas.

[74] JOHNSON, Selmer Martin. Optimal two-and three-stage production schedules with setup times included. En: Naval research logistics quarterly, 1954, vol. 1, no 1, p. 61-68.

[75] MUTH, John F.; THOMPSON, Gerald Luther. Industrial scheduling. Prentice-Hall, 1963.

Según Najid *et al*. (2001), son Brucker & Schlie[76](1990) quienes abordan por primera vez el Job Shop flexible. Ellos propusieron un algoritmo polinomial para resolver un FJSSP que contiene 2 trabajos. A partir de ahí, más estudios han sido realizados sobre este tema. También, Jansen *et al*. (2000), concluye que la complejidad del Job Shop flexible es mayor al Job Shop debido a incluir la necesidad de asignar operaciones a las maquinas.

Durante su estudio, Fattahi, Mehrabad & Jolai (2007), examinan que existen dos enfoques para solucionar un problema de *Job Shop* flexible: Un planteamiento jerárquico y un planteamiento integrado. En el primer caso, se trabaja el problema de asignación de máquinas como un problema aparte de la secuenciación; y en el segundo, se abordan ambos problemas de manera simultánea.

La descomposición es una de las posibles formas de solución cuando se tratan más de 2 trabajos (conocida como planteamiento jerárquico o por niveles), en el cual se descompone el *Job Shop* flexible en sub-problemas para así reducir su dificultad. Este método de descomposición fue trabajado por primera vez por Brandimarte[77] en 1993, quien resolvió el problema de asignación utilizando reglas de despacho (asemejándose al problema de ruteo) para luego aplicar búsqueda tabú sobre este y así resolver el problema de secuenciación (*Job Shop* clásico). Esta metodología también fue aplicada por Paulli J. (1995), Chambers J.B. & Barnes J.W. (1996), Kacem, Hammadi & Borne (2002), y Low, Yip & Wu (2006).

Por otra parte, Fattahi *et al*. (2007) exponen que también es usado el planteamiento integrado, el cual a diferencia del anterior, considera la asignación y la secuenciación al mismo tiempo. Se considera de mayor dificultad para ser resuelto,

[76] BRUCKER, Peter; SCHLIE, Rainer. Job-shop scheduling with multi-purpose machines. En: Computing, 1990, vol. 45, no 4, p. 369-375.

[77] BRANDIMARTE, Paolo. Routing and scheduling in a flexible job shop by tabu search. En: Annals of Operations research, 1993, vol. 41, no 3, p. 157-183.

pero según han descrito Vaessens *et al*. (1994), Hurink J, Jurish B, Thole M., (1994), Mastrolilli M & Gambardella LM., (1996) y Dauzére-Perés S & Paulli J., (1997) quienes lo usan, este enfoque ha dado mejores resultados.

Dentro de lo planteado por Zhang *et al*. (2012), sea ha expuesto que el *Job Shop* flexible se pueden encontrar en 3 tipos de sistemas de producción donde se involucra transporte: Sistemas flexibles de manufactura (FMS), Células robóticas de producción (RC) y superficies de tratamiento (STF); aunque esta última es tratada en el *Hoist scheduling problem* (Problema de programación de montacarga).

Según estos mismos autores, el problema de programación con restricciones de transporte ha sido estudiado solamente en casos de tamaño pequeño. En 2001, Hurink & Knust[78], solucionan un problema de *Flow shop* con tiempos de transporte y un solo recurso para esta tarea, teniendo como objetivo una programación factible con el mínimo *makespan*.

Durante los últimos años, se han desarrollado distintos estudios sobre el problema del *Job Shop* flexible. Algunos de los investigadores destacados exponen lo siguiente:

- Sankar (2003) examina la optimización multiobjetivo de un sistema de manufactura flexible (*FMS*) usando un Algoritmo Genético con dos estilos de codificación llamados *Pheno Style* y *Binary Style* con el fin de minimizar la tardanza de los trabajos y maximizar la utilización de las máquinas. Al final, se comparan los resultados obtenidos a través de la aplicación del Algoritmo Genético, con diferentes reglas de programación aplicadas al mismo problema tales como: SPT (*Shortest Procesing Time*), EDD (*Earliest Due Date*), SBQ

[78] HURINK, Johann; KNUST, Sigrid. Makespan minimization for flow-shop problems with transportation times and a single robot. En: Discrete Applied Mathematics, 2001, vol. 112, no 1, p. 199-216.

(*Standard Batch Quantity*); entre otros, concluyendo al Algoritmo Genético como la mejor opción en este problema.

- Hurink & Knust (2005), integran al problema de *Job Shop* las restricciones de transporte por medio de un robot que realice operaciones de movimientos entre máquinas, involucrando tiempos de movimientos llenos (transportando algún trabajo) y vacíos (cuando se desplaza sin carga entre las maquinas). El objetivo es minimizar el *Makespan*, para lo que presentan un algoritmo de búsqueda Tabú.

- Xia & Wu (2005) estudian un algoritmo hibrido usando optimización por enjambre de partículas (PSO) y recocido simulado (SA) para optimizar la programación de un *Job Shop* flexible con objetivos múltiples.

- Gao *et al*.(2006) estudia el problema de restricciones no fijas para la programación de un *Job Shop* flexible (*FJSSP-nfa*), el cual se considera una variación más real del *Job Shop* Flexible clásico, puesto que asume cada máquina sujeta a un número arbitrario de tareas asociadas a mantenimiento preventivo (PM). Por esta razón, proponen un algoritmo genético hibrido (hGA) en el cual sus operadores de cruce y de mutación son implementados en el espacio del fenotipo con el fin de incrementar su capacidad de heredar. Así mismo se trabaja con dos tipos de vecindarios basados en el concepto de la ruta crítica. Por último, un procedimiento de búsqueda local en los dos vecindarios definidos, donde se hibrida con el algoritmo genético para que represente un incremento en la habilidad de la búsqueda.

- Deroussi, Gourgand & Tchernev (2008), consideran un problema de programación *Job Shop* con vehículos de transporte (AGV), solucionado por medio de un sistema simple de búsqueda de vecindario eficiente, implementado

en tres distintas metaheurísticas: Iteración de búsqueda local (*Iterated local search*), recocido simulado (*simulated annealing*) y el hibrido de estos mismos.

- Zhang *et al*. (2009) proponen un algoritmo hibrido entre el PSO y la búsqueda tabú (TS), para solucionar un problema de objetivos múltiples del *Job Shop* flexible. Estos investigadores apuntan que el PSO muestra una alta eficiencia en la búsqueda al combinar búsquedas globales y locales.

- De Giovanni *et al*. (2010) presenta un algoritmo genético mejorado (IGA) para el problema de programación del *Job Shop* flexible Distribuido (DFJSP), donde los trabajos son procesados por un sistema de varias unidades de manufactura flexible (FMUs) generando así tareas adicionales al *Job Shop* flexible (realizado en cada uno de los FMU) tales como la asignación y secuenciación del trabajo en la FMU correspondiente y por consiguiente un *makespan* global el cual se buscará minimizar. Se presenta una solución por medio del algoritmo genético mejorado puesto que es necesario un ajuste en la codificación genética para incluir la información necesaria para declarar el problema *Job Shop* flexible distribuido.

- Baghery *et al*. (2010) proponen un algoritmo inmune artificial (AIA) como solución del problema de programación del *Job Shop* flexible teniendo como objetivo la minimización del *makespan*. Adicionalmente, diferentes estrategias fueron usadas para la generación de la población inicial. Estas estrategias son versiones modificadas del enfoque de localización propuesto por Kacem et al (2002).

- Zhang, Q *et al*. (2011) presentan un algoritmo genético con búsqueda tabú (*GATS*) para solucionar un problema de *Job Shop* flexible con restricciones de transporte y tiempos acotados de procesamiento (*Bounded processing times*), donde se trabajan dos objetivos: el primer minimizar el *makespan*, y el segundo,

minimizar el almacenamiento; que se maneja en términos de tiempo de espera de cada trabajo. En el método de solución, estos autores proponen la búsqueda tabú (TS) como parte del algoritmo genético, optimizando la secuencia de cada individuo de la población antes de ser seleccionados y cruzados.

- Moslehi & Mahnam (2011) examinan la optimización de la programación de un *Job Shop* flexible de objetivos múltiples a través de una metaheurística hibrida entre PSO (basada en prioridades) y búsqueda local; y utilizando el enfoque de frente de pareto, donde los objetivos son representados como componentes de un vector a optimizar.

- Tang *et al*. (2011) proponen una metaheurística al combinar el algoritmo genético con la optimización por enjambre de partículas para solucionar un problema de *Job Shop* flexible, y haciendo uso de operadores genéticos para mejorar el tiempo de ejecución del método en comparación con otros.

- Brucker *et al*. (2012) aplican un algoritmo *Branch and Bound* para el problema del *Job Shop* cíclico con transporte para minimizar el tiempo de ciclo. Estos autores desarrollan un método de búsqueda de árbol (*tree search method*) con un procedimiento de acotamiento, para compararlo en problemas resueltos por programación lineal.

- González et al. (2013) estudian el *Job Shop* flexible con tiempos de alistamiento (SDST-FJSP) por medio de un algoritmo memético, donde proponen una estructura de vecindario (*fast neighborhood structure*) compuesta por dos movimientos (*reversal crititcal arcs y machines assignment*) para hacer estimaciones más rápidas.

- Liu et al. (2013) presentan un algoritmo microartificial de colonia de abejas (MMABC) para solucionar un *Job Shop* flexible multiobjetivo con restricciones

de transporte, donde proponen un operador de cruce (*Crossover operator*) propuesto por Jun-Qing (2012) para mejorar el resultado de la programación, optimizando el *makespan* y el tiempo de procesamiento de cada operación, puesto que este varía en cada una de las máquinas.

- Lacomme *et al*. (2013) examinan la optimización del *makespan* en un *Job Shop* con vehículos guiados automatizados (AVGs) por medio de una estructura (*framework*) basado en un grafo disyuntivo para modelar el problema de forma conjunta con un algoritmo memético que solucione la programación de las máquinas y los vehículos.

- Rossi (2014) propone una solución basada en un planteamiento por niveles (*assigning and sequencing sub-problems*) para solucionar un problema de programación de *Job Shop* flexible con tiempos de transporte y tiempos de configuración dependientes de la secuencia (*sequence-dependent setup times*). El método expuesto es una metaheurística de colonia de hormigas reforzado con relaciones de feromonas (*Ant colony with reinforced pheromone relationships*). Es decir, un método basado en un enjambre inteligente (*swarm intelligence*) que utiliza conceptos de un modelo de grafo disyuntivo.

Según se observa, muchos autores han estudiado el problema del *Job Shop* flexible, pero en la mayoría de los casos, el transporte entre maquinas no es tenido en cuenta. A partir de lo dicho por Deroussi & Norre[79](2010), esta nueva modificación en la línea de investigación que considera el transporte como una operación a realizar con recursos para la misma, ha tomado auge en los últimos años por lo que esta investigación se centra en estudiar este tema en particular.

[79] DEROUSSI, L.; NORRE, S. Simultaneous scheduling of machines and vehicles for the flexible job shop problem. En: International conference on metaheuristics and nature inspired computing. 2010.

6. MARCO DE ANTECEDENTES

En Colombia, Castrillon, Sarache y Giraldo (2011) realizaron un proyecto investigativo en un ambiente *Job Shop* buscando la disminución del tiempo total de proceso (*makespan*) y el incremento del tiempo de trabajo de las máquinas por medio de una heurística basada en la optimización de colonia de hormigas (ACO). Los autores determinan que la optimización de colonia de hormigas (ACO) es una de las mejores soluciones para el proceso de secuenciación en un ambiente *Job shop* pues se obtuvo un 99% de aproximación a la solución óptima, siendo un punto de referencia para comparar la eficiencia de los diferentes métodos heurísticos a aplicar en este tema. Así mismo, para futuros estudios, se resalta la importancia de tener en cuenta cualquier variación en el *makespan* y el tiempo de inactividad para la obtención de excelentes resultados.

Estos autores, adicional a su actividad de desarrollo en investigación, llevaron a cabo un estudio en el cual realización la aplicación de un algoritmo evolutivo en la solución de problemas *Job Shop* de forma práctica para aumentar la utilización de máquinas en una empresa del sector metalmecánico. Castrillon *et al*. (2011) desarrollaron esta metodología de forma que fuera fácil de replicar dentro de la empresa, proponiendo un modelo. A lo largo de la investigación, trabajaron como función *fitness* del algoritmo evolutivo el tiempo muerto (*idle time*) y el *makespan*; pero observaron que este segundo objetivo era el que guiaba la solución, por lo que su enfoque posterior se basó solo en este. Estos autores observaron que al aplicar a la industria este tipo de investigaciones, lograron disminuir el tiempo total de proceso en un 33%, debido a que las técnicas usadas tradicionalmente (procesar de acuerdo a la operación más larga, orden de llegada u operación más corta) no tienen una base teórica para ser consideradas. En este proyecto de investigación, financiado por la Universidad Nacional de Colombia, los investigadores observaron cómo la aplicación de una herramienta robusta basada en un problema de estudio mejora los resultados de la industria, lo cual exponen como la razón última de la

investigación. Resaltando lo importante de estas investigaciones, se puede observar el enfoque en la minimización del *makespan* en ambos estudios, objetivo a tratar en el presente documento.

En Chile, Durán, Rojas & Daza presentan en el año 2011 una investigación relacionado en mayor proporción con el trabajo a desarrollar, pues abordan el *Job Shop flexible* con el mismo método de optimización. El proyecto investigativo aplica computacionalmente un algoritmo genético secuencial para resolver el problema del *Job Shop* flexible. Durante el desarrollo del mismo, los autores destacan la importancia de hacer un análisis para obtener los mejores óptimos en cuanto a la definición de los parámetros, los cuales generan una dependencia con el desempeño de la metaheurística. En este caso, se realiza un diseño experimental para definir los parámetros correspondientes. De igual manera, los autores concluyen que las soluciones podrían ser mejoradas diversificando la población inicial con operadores genéticos, aspecto que se tendrá en cuenta para el desarrollo del presente proyecto.

A nivel de la Universidad Industrial de Santander, se han desarrollado 2 proyectos relacionados con el *Job Shop*. En el primero de estos, Sarmiento (2012) presenta un algoritmo MOPSO para solucionar un problema *Job Shop* de múltiples objetivos, haciendo uso del frente de Pareto, una composición vectorial de los objetivos a tratar, que permite determinar soluciones no dominadas. Después de esta investigación, Aguilar y Pérez (2013) exponen un algoritmo memético para la minimización del *makespan*, donde determinan que los niveles del factor porcentaje de mutación no tienen efecto significativo sobre la variable respuesta, y a su vez, concluyen que estos algoritmos de estrategia genética son competitivos para problemas de secuenciación de alta complejidad.

Partiendo de estos estudios realizados en Colombia, dentro de la Universidad Industrial de Santander y el grupo OPALO, se observa la necesidad de seguir

avanzando en esta línea de investigación, atendiendo las recomendaciones propuestas por estudios anteriores dentro del grupo de investigación.

7. FLEXIBLE JOB SHOP SCHEDULING CON RESTRICCIONES DE TRANSPORTE

Adicional a lo propuesto en el *Job Shop* flexible, se hace necesario involucrar el transporte como una actividad adicional con recursos propios, desagregándolo para asignar recursos y poder asemejar el problema al presentado en la industria.

El propósito del problema es realizar la asignación de cada operación a cada una de las maquinas disponibles para estas, y adicionalmente, programar cada máquina de manera que se optimice el objetivo, en este caso el *makespan*.[80]

Junto a esto, al finalizar una máquina la operación, el trabajo debe ser transportado a otra, por lo que se hace necesario contar con recursos para esta actividad. Para esto, en el modelo se tienen *r* recursos idénticos de transporte encargados de las tareas de desplazamiento entre máquinas, siendo el tiempo de transporte dependiente de la trayectoria entre las máquinas.

Adicional a las consideraciones anteriores, en el modelo planteado también se tuvo en cuenta los tiempos determinístico de procesamiento de transporte en cada uno de los recursos conocidos.

[80] ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. En: Computers & Operations Research, 2012, vol. 39, no 7, p. 1713-1723.

7.1. FORMULACIÓN

En concordancia con lo dicho por Gallego *et al*.[81] (2012) y con lo revisado en la literatura, el *makespan*, definido como $max\{C_i\}$, donde C_i es el tiempo de finalización del trabajo *i*; es el objetivo a optimizar más utilizado, pues equivale a minimizar tiempos muertos o maximizar la utilización de las maquinas; por lo cual será el que se tendrá en cuenta para construir el modelo.

Finalmente, usando la nomenclatura que emplean Zhang *et al*.[82] (2012), se puede modelar el problema de la siguiente manera:

El modelo matemático se compone de una función objetivo y restricciones lineales, presentando variables de decisión para la asignación de recursos a las distintas operaciones.

Parámetros

J_i Trabajo i $(i \epsilon [1, n])$.

O_i Número de operaciones para el trabajo J_i $(i \epsilon [1, n])$.

OP_{ij} j esima operación del trabajo J_i $(j \epsilon [1, O_i])$.

P_{ijk} Tiempo de procesamiento para cada OP_{ij} en la maquina k.

C_i Tiempo de terminación del trabajo J_i.

M_k Máquina k $(k \epsilon [1, m])$.

MP_{ij} Conjunto de máquinas que pueden procesar la operación OP_{ij}

R_h Recurso de transporte $(h \epsilon [1, r])$.

TP_{ij} Asignación de trasporte entre OP_{ij} y OP_{ij+1}

[81] GALLEGO, Mario César Vélez; ZULUAGA, Carlos Alberto Castro; TORO, Jairo Maya. Algoritmo de búsqueda aleatoria para la programación de la producción en un taller de fabricación. En: Revista Universidad EAFIT, 2012, vol. 39, no 131, p. 76-86.

[82] Ibíd.

TR_{ij} Conjunto total de recursos de transporte que pueden asignarse a TP_{ij}.

σ_{kl} Tiempo de transporte vacío entre la máquina M_k y M_l $(k, l \epsilon [1, m])$.

τ_{kl} Tiempo de transporte con carga entre la máquina M_k y M_l $(k, l \epsilon [1, m])$.

Variables

t_{ij} Fecha de inicio de la OP_{ij} $(i \epsilon [1, n])$ $(j \epsilon [1, O_i])$.

v_{ij} Fecha de inicio para TP_{ij} $(i \epsilon [1, n])$ $(j \epsilon [1, O_i - 1])$.

PJ_{ijk} 1 si la OP_{ij} es realizada por la maquina M_k. $M_k \epsilon MP_{ij}$;

0 de lo contrario

$YP_{iji'j'k}$ 1 si la OP_{ij} es realizada antes de $OP_{i'j'}$ en la maquina *k*;

0 de lo contrario. (Dado que $(PJ_{ijk} = 1 \ PJ_{i'j'k} = 1)$).

TJ_{ijh} 1 si el TP_{ij} es realizado por el recurso R_h. $(h \epsilon [1, r])$;

0 de lo contrario

$YT_{iji'j'h}$ 1 si el TP_{ij} es realizado antes de $TP_{i'j'}$;

0 de lo contrario. (Dado que $TJ_{ijh} = 1 \ TJ_{i'j'h} = 1$)).

Función Objetivo *Minimizar* $C_{max} = max_{i \epsilon [1,n]} \{C_i\}$

Donde $C_i = t_{iO_i} + P_{iO_i}$

Restricciones

- Las restricciones de precedencia entre dos tareas de procesamiento en el mismo puesto de trabajo:

$$t_{ij} + P_{ijk} \leq t_{ij'}$$

- Cada operación de procesamiento debe ser asignada a uno y solo un recurso de procesamiento:

$$\sum_{k \epsilon PR_{ij}} PJ_{ijk} = 1$$

- Cada tarea de transporte debe ser asignada a uno y solo un recurso de transporte:

$$\sum_{h \in [1,r]} TJ_{ijh} = 1$$

- Restricción de capacidad para cada recurso de procesamiento. $\forall i, i' \in [1,n]$; $\forall j, j' \in [1, O_i - 1]$; $\forall k \ MP_{ij} \cap MP_{i'j'}$. Siendo *Q* un valor extremadamente grande

$$t_{ij} + P_{ijk} \leq t_{i'j'} + \left(1 - YP_{iji'j'k}\right) * Q$$

$$PJ_{ijk} * PJ_{i'j'k} * (t_{i'j'} + P_{i'j'k}) \leq t_{ij} + YP_{iji'j'k} * Q$$

  Es decir, una máquina no podrá procesar más de una operación al tiempo.

- Restricción de capacidad para cada recurso de transporte. $\forall i, i' \in [1,n]$, $\forall j \in [1, O_i - 1]$, $\forall j' \in [1, O_{i'} - 1]$ $\forall k \ TR_{ij} \cap TR_{i'j'}$, $\forall k, l, h \in [1,r]$.

$$v_{ij} + \sigma_{kl} + \tau_{kl} \leq v_{i'j'} + \left(1 - YT_{iji'j'h}\right) * Q$$

$$TJ_{ijh} * TJ_{i'j'h} * (v_{i'j'} + \sigma_{kl} + \tau_{kl}) \leq v_{ij} + YT_{iji'j'h} * Q$$

  Es decir, un recurso de transporte no podrá transportar más de una operación al tiempo.

- Restricción de transporte (un recurso de transporte debe tener tiempo libre para moverse entre dos operaciones sucesivas).

$$\forall i \in [1,n], \forall j \in [1, O_i - 1], \forall h \in [1,r], \forall k \in MP_{ij}, \forall k' \in MP_{ij+1}.$$

$$t_{ij} + P_{ijk} + \tau_{kk'h} \leq t_{ij+1}$$

8. ALGORITMO GENÉTICO PROPUESTO APLICADO AL FLEXIBLE JOB SHOP

Basado en el algoritmo hibrido expuesto por Zhang *et al*.[83] (2012) para dar solución al problema de FJSSP con ventanas de tiempo de procesamiento y recursos de transporte, se propone una algoritmo genético con modificaciones para abarcar el problema de *Job Shop* flexible con restricciones de transporte.

A continuación, se procederá a describir en detalle la programación propuesta.

8.1. PARÁMETROS DE ENTRADA

Para comenzar, se deben definir los parámetros de entrada que definen tanto al problema cómo al algoritmo. El problema del Job Shop flexible consiste en planear y organizar un conjunto de trabajos j que deben procesarse en un conjunto de máquinas m con restricción de operación, es decir, se presenta la necesidad de asignar las operaciones dependiendo de la posibilidad de procesamiento por parte de la máquina[84]. Con base en la anterior definición, el *Job Shop* flexible se compone de los siguientes parámetros:

- Número de trabajos: Cantidad de trabajos que llegan al sistema, para ser procesados en las diferentes máquinas.

- Número de operaciones: Cantidad de operaciones que conforman un trabajo.

[83] ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. En: Computers & Operations Research, 2012, vol. 39, no 7, p. 1713-1723.

[84] CANSEN, M. MASTROLILLI, R. Solis-Oba. Approximation algorithms for flexible job shop problems. Proceedings of Latin America Theoretical informatics. (LATIN 2000) LNCS 1776. pp. 68 - 77.

- Número de máquinas: Cantidad de máquinas en las que podrán ser procesadas determinadas operaciones.

- Matriz de tiempos de procesamiento: Tiempos de procesamiento de las operaciones en cada una de las máquinas. Teniendo en cuenta que el criterio establecido es el *makespan* y con base en el artículo "*A genetic algorithm for the Flexible Job Shop problem*" de Durán[85], la matriz se compone de operaciones (filas) y máquinas (columnas) de manera que, cuando la máquina no puede procesar una determinada operación, se asigna un tiempo de procesamiento muy grande (M) con el fin de impedir dicha asignación (Figura 19).

Figura 19. Ejemplo Matriz de Tiempos de Procesamiento

	M_1	M_2	M_3
$O_{1,1}$	1	2	1
$O_{1,2}$	M	1	1
$O_{1,3}$	4	3	M
$O_{2,1}$	5	M	2
$O_{2,2}$	M	2	M
$O_{2,3}$	7	5	3

- Matriz de tiempos de transporte: Establece los tiempos de transporte entre máquinas, por lo cual, la matriz se define como se presenta en la figura 20.

[85] MEDINA DURÁN, R., PRADENAS ROJAS, L., & PARADA DAZA, V. Un algoritmo genético para el problema de Job Shop Flexible. Ingeniare. Revista chilena de ingeniería, 2011, 19(1), 53-61.

Figura 20. Ejemplo Matriz de Tiempos de Transporte

FMS1	LU	M_1	M_2	M_3	M_4
LU	0	6	8	10	12
M_1	12	0	6	8	10
M_2	10	6	0	6	8
M_3	8	8	6	0	6
M_4	6	10	8	6	0

Por otro lado, como se expuso en el capítulo 2.5, el algoritmo genético propuesto por Holland[86], se describe como “Algoritmo de búsqueda basado en la mecánica de selección natural y de la genética natural. Combina la supervivencia del más apto entre estructuras de secuencias con intercambio de información estructurado, aunque aleatorizado, para construir así un algoritmo basado en la genialidad de la búsqueda humana” compuesto por los siguientes parámetros iniciales: número de generaciones, tamaño de la población, probabilidad de cruce y probabilidad de mutación.

- Número de Generaciones: hace referencia al número de veces que se corre el algoritmo en su totalidad (evaluación, selección, cruce, mutación).

- Tamaño de población: Cantidad de individuos, cromosomas o soluciones que ingresan a iterar en cada una de las generaciones.

- Probabilidad de Cruce: Como se mencionó en la sección X, la probabilidad de cruce determina la ocurrencia del mismo. El procedimiento consiste en generar un número aleatorio entre 1 y 100, sí este número está entre 1 y la probabilidad

[86] HOLLAND. Op. cit.

de cruce, se procede con el cruce respectivo. De lo contrario, no se procede a cruzar los cromosomas.

- Probabilidad de Mutación: Se repite el procedimiento mencionado en el ítem anterior para determinar la realización de la mutación.

8.2. CODIFICACIÓN PROPUESTA

El algoritmo presentado comienza generando una población, la cual está compuesta por individuos denominados cromosomas. Un cromosoma está compuesto por genes que describen la asignación de una tarea en una máquina, y a su vez, cada uno de ellos cuenta con un similar donde se encuentra almacenada la información de la asignación de máquinas, siendo en conjunto la secuencia de tareas con su respectiva asignación una solución factible para el problema del Job Shop Flexible, cumpliendo con las restricciones descritas en la formulación y teniendo en cuenta las siguientes consideraciones[87]:

- Precedencia entre dos tareas de procesamiento del mismo trabajo, es decir, que las tareas se realizan con el orden indicado.

- Cada operación de procesamiento debe ser asignada a un solo recurso de procesamiento.
- Restricción de capacidad para cada una de las máquinas, teniendo en cuenta que al problema ser de tipo flexible, las capacidades de procesamiento pueden variar de una máquina a otra.

[87] ZHANG, Q.; MANIER, H.; MANIER, M. A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. Computers & Operations Research, 39(7), 1713-1723, 2012.

Para explicar con mayor detalle cómo se abarcó el problema desde la perspectiva de esta metaheurística, se partirá de un ejemplo donde se cuenta con 3 trabajos (J_1, J_2, J_3), 4 Maquinas (M_1, M_2, M_3, M_4) y un recurso de transporte (R_1):

J_1: $TP_{L/U}$; OP_{11} (M_1,M_2); TP_{11} ; OP_{12} (M_3,M_4); TP_{12} ; OP_{13}(M_2,M_3).

J_2: $TP_{L/U}$; OP_{21} (M_1,M_2); TP_{21} ; OP_{22} (M_2,M_3); TP_{22} ; OP_{23}(M_2,M_4).

J_3: $TP_{L/U}$; OP_{31} (M_2,M_4); TP_{31} ; OP_{32} (M_1,M_3); TP_{32} ; OP_{33}(M_1,M_4).

La lectura del problema puede realizarse de la siguiente manera: Se tienen tres trabajos por realizar, cada uno compuesto por 3 tareas u operaciones. Para el trabajo uno, se tiene que la operación 1 puede ser procesada por la maquina 1 o la maquina 2. Posteriormente, se debe realizar un transporte TP_{11} para llevar el trabajo de la maquina donde se realice la operación 1 hasta la maquina donde se realice la operación 2, y así sucesivamente. De igual manera se presenta el $TP_{L/U}$ el cual es el transporte del material a procesar desde la “bodega” a la máquina respectiva (movimiento inicial).

Como se observa, para poder asignar los transportes, se debe resolver inicialmente el problema de asignación, puesto que el tiempo de transporte depende de las máquinas, atadas a una configuración dentro de la planta o *layout*; por lo que el proceso de solución es el siguiente:

- Resolver el problema de secuenciación de operaciones.
- Resolver el problema de asignación de cada una de las operaciones de procesamiento en las respectivas máquinas.
- Incluir la información de tiempos de transporte partiendo de la asignación hecha de las operaciones de transporte, así como la inclusión de los mismos en el proceso de iteración dentro del algoritmo.
- Deducir la asignación de operaciones de transporte en los respectivos recursos dedicados a esta actividad.

Cromosoma

La estructura de los genes que conforman el cromosoma es clave para el desarrollo de la codificación planteada. Con base en los diferentes tipos de representaciones genéticas para el FJSSP, en esta investigación se usa la representación basada en operaciones codificando un Schedule por medio de una secuencia de operaciones. Lo anterior fue adaptado de acuerdo a lo propuesto por Gao et al (2007).[88]

Con base en el ejemplo planteado anteriormente, un cromosoma cualquiera sería [3 1 2 3 1 2 3 1 2], el cual se lee de izquierda a derecha y donde (1, 2 y 3) corresponden a los trabajos J_1, J_2 y J_3, y las repeticiones representan las operaciones de cada uno de los trabajos respectivamente. El primer número de cada trabajo corresponde a la primera tarea respectivamente. La repetición de números corresponde a la secuencia de operaciones de cada trabajo de manera ascendente. De este modo, siempre se respetará la restricción de precedencia entre operaciones asegurando la factibilidad del cromosoma.

En un cromosoma se encontrarán repeticiones según el mayor número de operaciones que se presenten en un trabajo. Al ser un FJSSP, las operaciones de cada uno de los trabajos pueden variar, para lo cual, en la matriz de tiempos determinada como parámetro inicial del problema, se asignará un tiempo de procesamiento cero según corresponda con el fin de no alterar el *makespan* resultante.

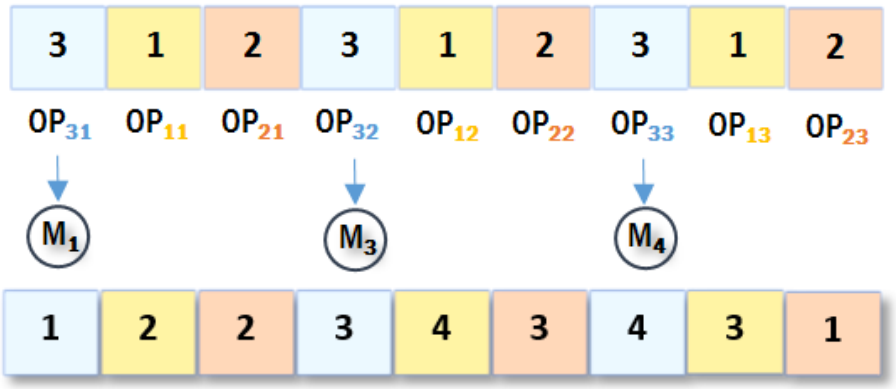
[88] GAO, Jie; SUN, Linyan; GEN, Mitsuo. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research, 2008, vol. 35, no 9, p. 2892-2907.

Figura 21. Interpretación del cromosoma.

Asignación de operaciones[89]

Una vez construido el cromosoma, se procede a asignar las operaciones en las máquinas. El criterio de asignación es aleatorio, dejando la optimización al algoritmo evolutivo.

Figura 22. Asignación de operaciones a las máquinas.



Esta información es almacenada en un cromosoma, donde en conjunto, ambos cromosomas representan una solución factible al problema.

Cabe aclarar que la información de asignación está ligada al cromosoma de secuenciación, por lo que al ejecutarse cualquier procedimiento de optimización o

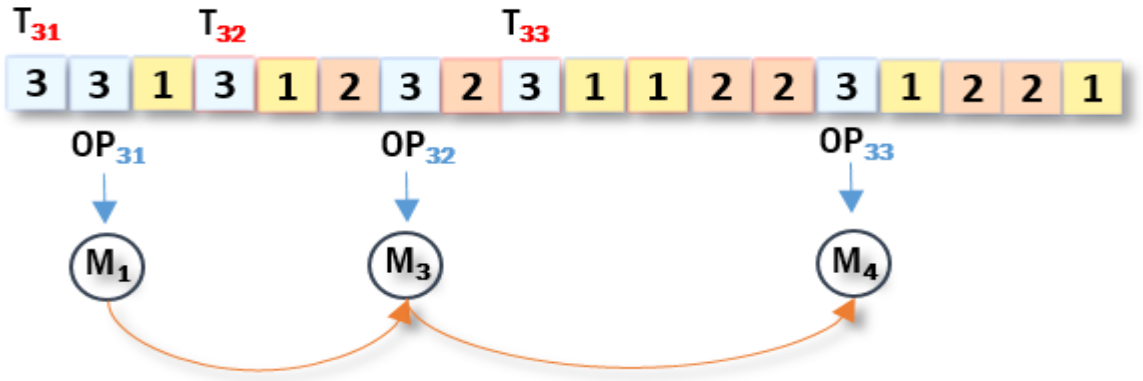
[89] GAO, J.; SUN, L.; GEN, M. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research, 2008, 35(9), 2892-2907.

etapa dentro del algoritmo (cruce o mutación), la información de asignación será transmitida en el algoritmo genético a cromosomas hijos.

Asignación de transporte

Adicional al procesamiento de trabajos, en la presente investigación se considera el transporte de las operaciones entre máquinas como parte integral del proceso de optimización. Por tal motivo, se propone integrar la asignación de máquinas con la asignación de transporte en un solo cromosoma como solución al problema planteado. Para lo anterior, el número de operaciones en cada trabajo se duplica al tener en cuenta que cada operación de procesamiento está precedida por una operación de transporte. Con esto, modificamos cada uno de los cromosomas que componen la población teniendo como resultado el cromosoma presentado en la Figura 23.

Figura 23. Asignación de transporte de operaciones entre máquinas.



Por lo tanto, los números impares de cada una de las repeticiones de números corresponden a los transportes de operaciones de bodega a máquina y de máquina a máquina según corresponda.

Generación de la población inicial

A pesar de la existencia de diferentes métodos para generar la población inicial como los presentados en el marco teórico, en la presente investigación la población

inicial es generada de forma aleatoria, de manera que el espacio de solución se explore sin tener en cuenta soluciones iniciales.

En la generación de la población, se debe tener presente el tamaño de la población a analizar, ya que no puede ser ni tan pequeña como para que se llegue prematuramente a un mínimo local, ni tan grande que comprometa más tiempo de solución computacional.

Selección

En cada iteración, el mejor cromosoma es escogido para una futura reproducción. Para la presente investigación, el operador de selección aplicado fue la selección por torneo, donde a partir de un porcentaje asignado, aleatoriamente se toma un subconjunto de la población en el cual se evalúa qué cromosoma es el mejor frente a los demás basándose en el *fitness* de cada uno, que para nuestro caso, es el *Makespan*. Dicho porcentaje de selección, definirá el tamaño del subconjunto. Entre mayor sea el porcentaje de selección, mayor será la probabilidad de escoger a los mejores individuos. Por el contrario, entre menor sea porcentaje de selección, la probabilidad de elegir a un individuo menos apto será mayor.[90]

Cruce

Con base en lo expuesto por De Jong[91] y los resultados de su artículo *"An analysis of the behaviour of a class of genetic adaptive systems"*, se concluye que el cruce basado en dos puntos es el que mejores resultados arroja en comparación con 3 o más puntos dado que a mayor cantidad de puntos de cruce se puede perder la

[90] MILLER, Brad L.; GOLDBERG, David E. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 1995, vol. 9, no 3, p. 193-212.

[91] DE JONG. Op.cit

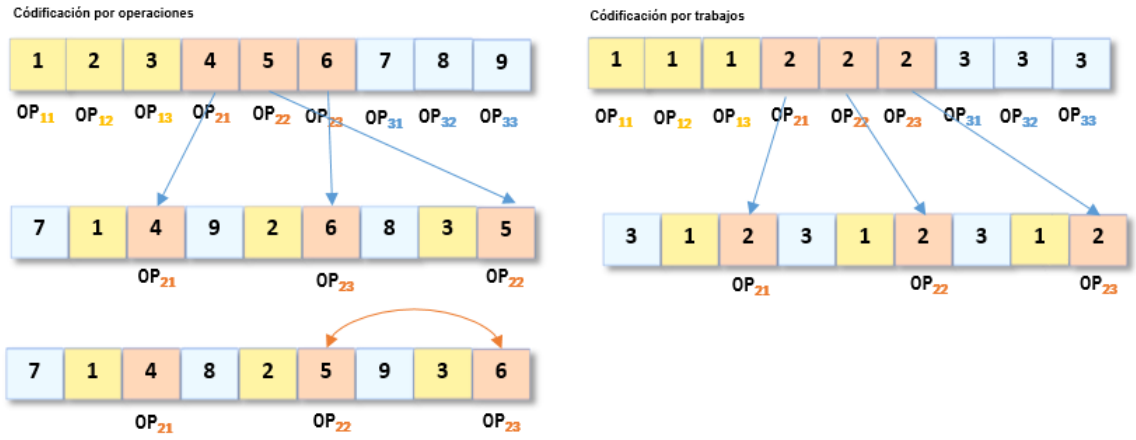
información genética, el operador de cruce utilizado en la presente investigación es el cruce de dos puntos.

Realizado con el fin de intercambiar la información genética y así combinar los cromosomas más aptos los cuales pasarán a la siguiente generación, el cruce de dos puntos escoge de manera aleatoria dos puntos los cuales indicarán los dos genes a cruzar (Ver Capitulo 2.5.1).

La probabilidad de cruce determinará la realización del cruce y por lo tanto definirá una mayor o menor exploración del espacio de soluciones. En primera instancia un número aleatorio entre 1-100 es generado y, cada vez que dicho número esté por debajo de la probabilidad definida, se ejecutará el cruce.

Gracias al tipo de codificación del cromosoma propuesto (ver página 73), no será necesario ejecutar lo que en el marco teórico llamamos “Reparación” como una revisión de la factibilidad del cromosoma en cuanto al secuenciación.

Figura 24. Comparación de codificaciones.

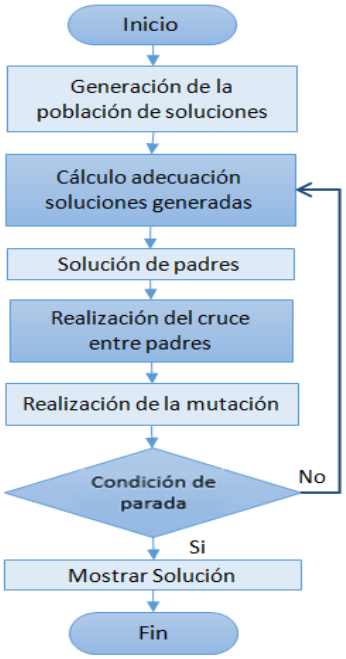


Antes de continuar con el proceso, se almacena el mejor valor encontrado de la población al evaluarlo nuevamente con la función *fitness,* con el fin de que no se pierda al ingresar a la siguiente etapa del algoritmo.

Mutación

La mutación corresponde al último paso del algoritmo genético, el cual tiene como objetivo la diversificación de la población cambiando de forma arbitraria algunos de los genes de un individuo dependiendo del tipo de mutación que sea escogido (ver 2.5.1). En la codificación propuesta, se ha escogido la mutación swap, en la cual se seleccionan de manera aleatoria dos genes para ser intercambiados uno por el otro. La probabilidad de mutación controla el porcentaje en el cual se introducen estos cambios en la población, por medio de la generación de un número aleatorio entre 0 -1 el cual, de ser menor o igual a la probabilidad de mutación, la mutación será ejecutada. De igual manera, por medio del tipo de codificación del cromosoma, se garantiza que la mutación no generará infactibilidad en el cromosoma.

Figura 25. Diagrama de flujo del AG.



Fuente. Adaptado de PALACIOS, Melissa; JAIMES, Christian. Alternativa de solución al problema de distribución de planta para instalaciones de áreas iguales y desiguales mediante un Algoritmo Hibrido Genético. Proyecto de grado. Bucaramanga: Universidad Industrial de Santander. Facultad de Ingeniería Físico-Mecánicas. 2011. 139 p.

9. VALIDACIÓN DEL ALGORITMO

Se denomina validación al proceso de verificación en la ejecución de un sistema en un ambiente específico. Con base en lo anterior, se validó el desempeño de la codificación propuesta, a través de la comparación de los resultados versus las instancias de diferentes etapas representadas por medio de la descomposición del Job Shop Flexible con recursos de transporte.

La información numérica fue obtenida a través de un seudocódigo hecho en lenguaje de programación Matlab®, ejecutado en un PC con procesador Intel core i7, 2 GHz y 8 Gb de memoria RAM.

Las etapas propuestas son:

- Problema del Job Shop
- Problema del Job Shop Flexible
- Problema del Job Shop con recursos de transporte
- Problema del Job Shop Flexible con recursos de transporte.

Las instancias a utilizar para validar el algoritmo generado fueron propuestas por Fisher & Thomson[92] y Lawrence[93] para validación en instancias de tipo *Job Shop*,

[92] FISHER, Henry; THOMPSON, Gerald L. Probabilistic learning combinations of local job-shop scheduling rules. *Industrial scheduling*, 1963, vol. 3, p. 225-251.

[93] LAWRENCE, S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (supplement). *Graduate School of Industrial Administration*, 1984.

Fattahi[94] y Paulli [95] para validación de tipo *Job Shop* flexible, y Deroussi[96] para validar el código en instancias de tipo *Job shop* y *Job Shop* flexible con restricciones de transporte.

Esta documentación se basó en las validaciones realizadas a lo largo de la construcción del seudocódigo, estableciendo finalmente un algoritmo con posibilidad de arrojar soluciones factibles para problemas con estas características. Para los distintos tipos de problemas a validar, la modificación a realizar sobre el seudocódigo se hace en los parámetros de entrada, de acuerdo a la información suministrada por cada una de las instancias.

9.1. RESULTADOS DE LA VALIDACIÓN PARA EL PROBLEMA DEL JOB SHOP

Por medio de las instancias desarrolladas en el trabajo "Algoritmo Memético para la minimización del *makespan* en el problema del *Job Shop scheduling*" realizado por Aguilar[97], procedemos a confrontar los resultados obtenidos y verificar la efectividad de la codificación propuesta en la presente investigación para problemas de tipo Job

[94] FATTAHI, Parviz; MEHRABAD, Mohammad Saidi; JOLAI, Fariborz. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. En: Journal of Intelligent Manufacturing, 2007, vol. 18, no 3, p. 331-342.

[95] DAUZÈRE-PÉRÈS, Stéphane; PAULLI, Jan. An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. *Annals of Operations Research*, 1997, vol. 70, p. 281-306.

[96] DEROUSSI, L. op. cit.

[97] AGUILAR, Karin y PÉREZ, Yuleiny. Un algoritmo memético para la minimización del makespan en el problema del Job shop scheduling. Bucaramanga, 2012, 223p.

Shop, a través de las instancias de Fisher & Thomson[98] y Lawrence[99]. De las instancias, se decidió tomar 10 de manera aleatoria.

Para ello, se realizó inicialmente la modificación de los parámetro de entrada, con base en lo establecido para el seudocódigo (ver numeral 8.1). Se modificó la matriz de tiempos de transporte a partir de valores nulos (ceros) y la matriz binaria que posibilita la asignación de máquinas a solo una.

Los resultados se muestran a continuación:

Tabla 1. Comparación entre los resultados encontrados con el Algoritmo Genético e instancias Job Shop.

Instancias de Job Shop					
Instancia	**Jobs**	**Máquinas**	**C MAX obtenido**	**Mejor CMAX**	**% Error**
Ft06	6	6	55	55	0,00%
Ft10	10	10	978	937	4,38%
Ft20	20	5	1217	1165	4,46%
La01	10	5	666	666	0,00%
La06	15	5	926	926	0,00%
La09	15	5	951	951	0,00%
La14	20	5	1292	1292	0,00%
La17	10	10	801	784	2,17%
La21	15	10	1079	1046	3,15%
La38	15	15	1254	1196	4,85%

Con base en las instancias presentadas y las tendencias identificadas en comparación con el algoritmo memético construido por Aguilar, se puede concluir que el Algoritmo genético con restricción de transporte optimiza el problema del Job Shop siempre y cuando se aumente el número de generaciones y el tamaño de la

[98] FISHER, Henry; THOMPSON, Gerald L. Op. cit

[99] LAWRENCE, S. op. cit.

población, ya que al no contar con heurísticas de búsqueda local, el algoritmo genético exige una mayor cantidad de procesos y tiempo de computo adicional para conseguir el mismo valor de *makespan*. El valor de los parámetros para ejecutar el algoritmo genético para la obtención de los resultados descritos en la tabla fueron los siguientes: Tamaño de población= 70. Generaciones= 1000. Porcentaje de mutación 9%. Porcentaje de cruce= 75%.

Las diferencias presentadas en las instancias de mayor complejidad se deben al procesamiento que debe realizar el algoritmo al estar diseñado para problemas con inclusión de transporte, que aun cuando son valores nulos, representan espacio de memoria, tiempo de solución y procedimientos adicionales dentro del algoritmo, que posibilitan la presencia de este margen de error cercano al 4%.

9.2. RESULTADOS DE LA VALIDACIÓN PARA EL PROBLEMA DEL JOB SHOP FLEXIBLE

A partir de las instancias presentadas en por Fattahi[100], Paulli[101] y Hurink, se realizó la validación de la codificación para problemas de tipo Job Shop flexible, en donde las operaciones pueden ser realizadas por más de una máquina. Al igual que en la validación anterior, se tomó una muestra de 12 instancias al azar, para validar los la efectividad del pseudocódigo en este tipo de problemas.

Para esta situación, los parámetros de entrada que no son relevantes dentro del algoritmo son los relacionados a tiempos de transporte. Al igual que en la validación del capítulo 9.1, este parámetro es establecido con valores nulos. Los resultados obtenidos para las instancias testeadas fueron los siguientes:

[100] FATTAHI, Parviz; MEHRABAD, Mohammad Saidi; JOLAI, Fariborz. Op cit.

[101] DAUZÈRE-PÉRÈS, Stéphane; PAULLI, Jan. op cit.

Tabla 2. Comparación entre los resultados encontrados con el Algoritmo Genético e instancias Job Shop Flexible.

Instancias de Job Shop flexible					
Instancia	**Jobs**	**Máquinas**	**C MAX obtenido**	**Mejor C_{MAX}**	**% Error**
Fattahi01	2	2	107	107	0,00%
Fattahi04	3	2	355	355	0,00%
Fattahi06	3	3	320	320	0,00%
Fattahi10	4	5	516	516	0,00%
Fattahi20	12	8	1237	1208	2,40%
Paulli 1	10	5	2616	2568	1,87%
Paulli 5	10	5	2279	2243	1,60%
Hurink01	6	6	47	47	0,00%
Hurink06	10	5	493	477	3,35%
Hurink10	15	5	766	750	2,13%
Hurink14	20	5	1096	1072	2,24%
Hurink23	10	10	810	757	7,00%

Por medio de la tabla 2, podemos concluir la efectividad del código de Job Shop Flexible con Restricciones de transporte para problemas de tipo *Job Shop* flexible de menor complejidad donde el porcentaje de error es de 0%. Como se puede observar, en los problemas de mayor complejidad como los son las instancias “Paulli 1” y “Fattahi 20”, obtenemos un porcentaje de error mínimo (cercano al 3%).

Dentro de la verificación realizada en el caso de las instancias propuestas por Hurink, se observa que dichos valores óptimos fueron obtenidos por medio de una metodología compuesta, donde se realiza un algoritmo hibrido en el cual cada metaheurística soluciona un respectivo problema, de asignación y de secuenciación. Aún así, se observa como el método utilizado no presenta errores superiores al 3,4%, solo para el caso atípico de la instancia final, donde su complejidad por el tamaño del problema deba ser abarca por métodos más avanzados.

9.3. RESULTADOS DE LA VALIDACIÓN PARA EL PROBLEMA DEL JOB SHOP CON RESTRICCIONES DE TRANSPORTE

Posteriormente, se validó el algoritmo programado a través de instancias propuestas por Deroussi[102], la cuales se caracterizan por ser problemas de tipo Job Shop con restricciones de transporte. Estas instancias se basan en una combinación de *Jobsets* y distribuciones de planta o *Layout* de donde se obtienen los tiempos de transporte entre máquinas. Para esta situación, todos los parámetros de entrada establecidos en el seudocódigo son utilizados según el problema planteado.

Los resultados obtenidos fueron los siguientes:

Tabla 3. Comparación entre los resultados encontrados con el Algoritmo Genético e instancias *Job Shop* con restricciones de transporte.

Instancias de Job Shop flexible con restricciones de transporte					
Instancia	C MAX obtenido	Algoritmo SALS	Algoritmo TSAG	Mejor CMAX encontrado	% Error
EX11	96	96	96	96	**0,00%**
EX12	86	82	82	82	4,88%
EX13	89	84	84	84	5,95%
EX21	106	100	104	100	6,00%
EX22	84	76	76	76	10,53%
EX23	95	86	86	86	10,47%
EX31	108	99	99	99	9,09%
EX32	92	85	85	85	8,24%
EX33	96	86	86	86	11,63%
EX41	117	112	116	112	4,46%
EX42	93	87	91	87	6,90%
EX43	98	89	94	89	10,11%
EX51	87	87	88	87	0,00%
EX52	74	69	69	69	7,25%

[102] DEROUSSI, L. op. Cit.

EX53	79	74	74	74	6,76%
EX61	124	118	120	118	5,08%
EX62	109	98	98	98	11,22%
EX63	112	103	103	103	8,74%
EX71	111	111	115	111	0,00%
EX72	82	79	85	79	3,80%
EX73	90	83	93	83	8,43%
EX81	163	161	161	161	1,24%
EX82	159	151	151	151	5,30%
EX83	166	153	153	153	8,50%
EX91	123	116	116	115	6,96%
EX92	111	102	102	102	8,82%
EX93	114	105	105	105	8,57%
EX101	153	147	148	147	4,08%
EX102	147	135	135	135	8,89%
EX103	153	138	141	138	10,87%

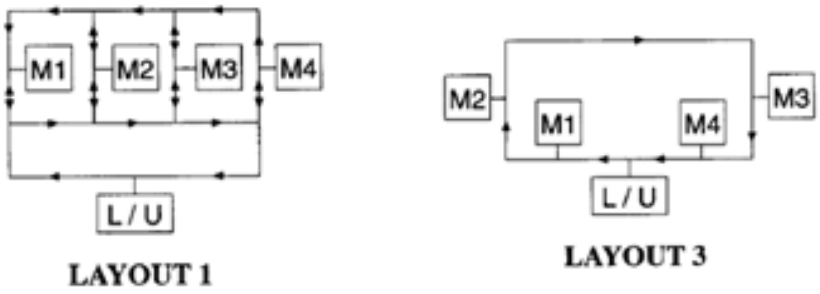
De las instancias solucionadas para validar el algoritmo, se puede observar que en tres de ella se coincidió con la mejor solución encontrada hasta el momento, e igualmente, en tres casos, se obtiene una mejor solución que los autores Zhang Manier[103] (TSAG).

Adicionalmente, se puede observar que se presenta un margen de error no mayor al 11,6%, que al observar en las distintas instancias, se presentan los mayores porcentajes de error en la configuración de planta “Layout 3” presentadas en la figura 26.

Esto se debe al tipo de configuración presentado en ese caso, donde no se tiene una simetría en los tiempos de desplazamiento entre máquinas, sino que existe una relación entre el tiempo de transporte y la dirección de desplazamiento, lo que hace más complejo el problema.

[103] ZHANG, H. MANIER, M. op cit.

Figura 26. Configuración de las plantas en las instancias propuestas por Deroussi.



Fuente. DEROUSSI, L.; NORRE, S. Simultaneous scheduling of machines and vehicles for the flexible job shop problem. En: International conference on metaheuristics and nature inspired computing. 2010.

Debido a esto, podemos concluir que para este tipo de configuraciones complejas, se hace necesario el apoyo al algoritmo a través de una heurística o algún método hibrido, tal y como ha sido desarrollado por otros autores, los cuales realizan una subdivisión del problema en asignación y secuenciación para estos casos, cada una con una metaheurística respectiva.

9.4. RESULTADOS DE LA VALIDACIÓN PARA EL PROBLEMA DEL JOB SHOP FLEXIBLE CON RESTRICCIONES DE TRANSPORTE.

Finalmente, se realizó la validación del algoritmo por medio de problemas de tipo flexible con restricciones de transporte. Para esto, se modificó las instancias de Deroussi[104] para obtener instancias de tipo Job Shop flexible, con flexibilidad parcial, de manera que al procesar dicha instancias se pueda observar una disminución en el valor del *makespan*.

De allí, se seleccionaron aleatoriamente las operaciones a flexibilizar y se asignó de la misma manera la máquina adicional que podría procesarla. Esta flexibilización de

[104] DEROUSSI, L. op cit.

instancias tiene como supuesto el hecho de que un tiempo de procesamiento es independiente de la máquina en la cual se procese la operación.

Las entradas obtenidas para la validación de esta etapa del algoritmo se adjuntan en el ANEXO B.

Las instancias fueron nombradas de la siguiente manera: EXF11, que hace referencia a la instancia original del autor, en composición al jobset *layout* predeterminados, pero introduciendo dentro de su notación la F, indicando que fueron modificadas para poder trabajarla de manera flexible.

Tabla 4. Comparación entre los resultados encontrados con el Algoritmo Genético e instancias Job Shop Flexible con restricciones de transporte.

Instancias de Job Shop flexible con restricciones de transporte					
Instancia	Valor obtenido	Valor óptimo sin flexibilizar	% Error	Valor encontrado sin flexibilizar	% Error
EXF12	76	82	7,89%	86	13,16%
EXF22	70	76	8,57%	84	20,00%
EXF32	80	85	6,25%	92	15,00%
EXF33	81	86	6,17%	98	20,99%
EXF43	85	89	4,71%	98	15,29%
EXF62	79	98	24,05%	109	37,97%
EXF82	116	151	30,17%	159	36,07%
EXF83	114	152	33,30%	166	45,61%
EXF81	118	161	36,44%	163	38,13%
EXF91	101	116	14,85%	123	21,78%

De la tabla se puede concluir que al flexibilizar las instancias, el algoritmo es capaz de realizar mejores asignaciones para obtener un menor valor de *makespan*. Si bien el porcentaje de mejora dependerá en gran parte del tipo de flexibilidad que se tenga en el problema (parcial o total), se puede observar que para la flexibilización del problema de forma aleatoria, se obtuvo una mejor solución como mínimo en un 4,71%.

9.5. ANÁLISIS DE RESULTADOS

Para poder evaluar el algoritmo y su comportamiento respecto a los 4 factores que indicen en su eficiencia para solución de problemas tipo *Job shop* flexible, se realizó un diseño de experimentos factorial 2^{k-1}, teniendo en cuenta como variable respuesta el *makespan*. Se tomaron 4 factores y dos niveles para cada uno de ellos, con 5 réplicas por cada uno de los tratamientos.

Para esto, se partió de la base establecida por Box et al (2005), donde se expone que deben ajustarse cada uno de los escenarios, realizando las ejecuciones del algoritmo de manera aleatoria después de haber definido el diseño factorial.

Finalmente, después de haber obtenido los resultados para las muestras de instancias, se procedió a analizar los resultados en el software estadístico Minitab. La información numérica con la cual se alimentó el diseño de experimentos elegido fue extraída de la formulación hecha en el lenguaje de programación Matlab, ejecutado en un PC con procesador Intel core i7, 2 GHz y 8 Gb de memoria RAM.

9.5.1. Instancias a testear

Teniendo en cuenta que el algoritmo propuesto fue desarrollado para solucionar problemas de tipo Job shop flexible con restricciones de transporte, los diseños de experimentos a ejecutar se basaron en las instancias propuestas por Deroussi[105], de las cuales, se seleccionaron 7 de las 32 posibles. Las instancias obtenidas para realizar los diseños fueron: EX11, EX31, EX32, EX43, EX62, EX82 y EX93.

9.5.2. Factores

Los factores analizados son los que presentan mayor influencia sobre la variable respuesta, basados en los parámetros de entrada del algoritmo, fueron identificados

[105] DEROUSSI, L. op cit.

los siguientes: Numero de generaciones, tamaño de población, porcentaje de selección y porcentaje de mutación.

Tabla 5. Factores tomados en el Análisis de Varianza.

Factores	Niveles	
	Bajo	Alto
Número de generaciones	100	900
Tamaño de población	10	100
Probabilidad de selección	0,01	0,1
Probabilidad de mutación	0,5	0,9

Adicionalmente, por ser el diseño factorial 2k-1, no se ejecutarán muestras de todos los posibles tratamientos, debido a que el diseño factorial elegido es fraccionado, así que solo se trabajarán 8 de los 16 posibles tratamientos.

Tabla 6. Definición de los niveles para el Análisis de Varianza.

% Mutación		% Selección		TamPob		Generaciones	
-	+	-	+	-	+	-	
0,01	0,1	0,5	0,9	10	100	100	900
x		x		x		x	
	x	x		x			x
x			x	x			x
	x		x	x		x	
x		x			x		x
	x	x			x	x	
x			x		x	x	
	x		x		x		x

9.5.3. Análisis de varianza para el problema EX11

Figura 27. Diagrama de Pareto de Efectos Estandarizados para el problema EX11.

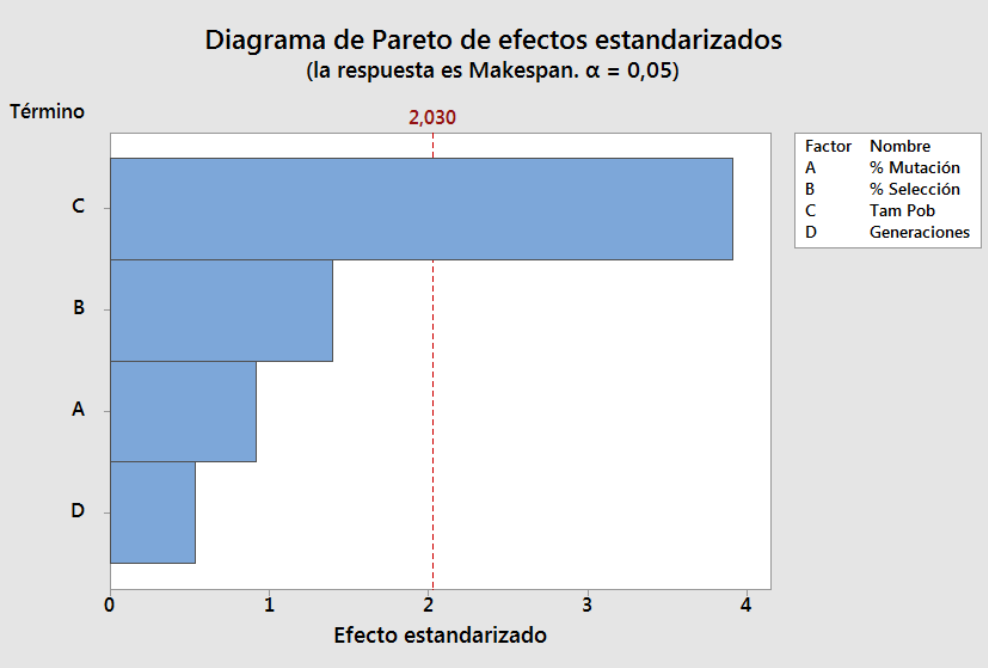


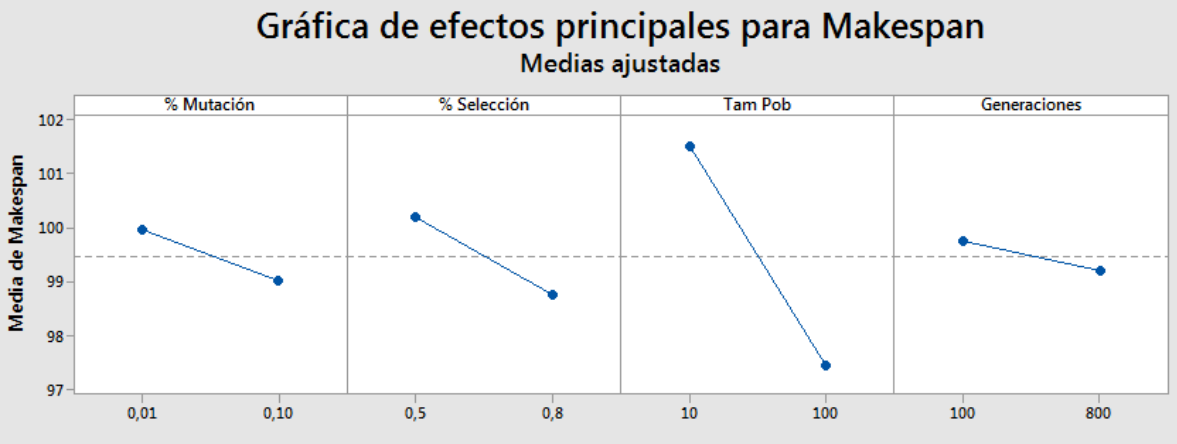
Tabla 7. Análisis de varianza para el problema EX11.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	9,025	9,025	0,84	0,365
% Selección	1	21,025	21,025	1,96	0,17
Tam Pob	1	164,025	164,025	15,31	0,0001
Generaciones	1	3,025	3,025	0,28	0,598
Error	35	374,875	10,711	14,14	
Total	39	571,975			

De acuerdo a lo obtenido en la tabla ANOVA y tal como se observa en la gráfica de efectos principales estandarizados para la instancia testeada, se puede concluir que solo el tamaño de población es significativo en este caso, los demás factores no son representativos en el modelo. Según la gráfica de efectos principales, se puede

afirmar que el *makespan* se minimiza cuando este factor (tamaño de población) se encuentra en su nivel alto.

Figura 28. Gráfica de Efectos principales para *Makespan* problema EX11.



9.5.4. Análisis de varianzas para el problema EX31

Figura 29. Diagrama de Pareto de Efectos Estandarizados para el problema EX31.

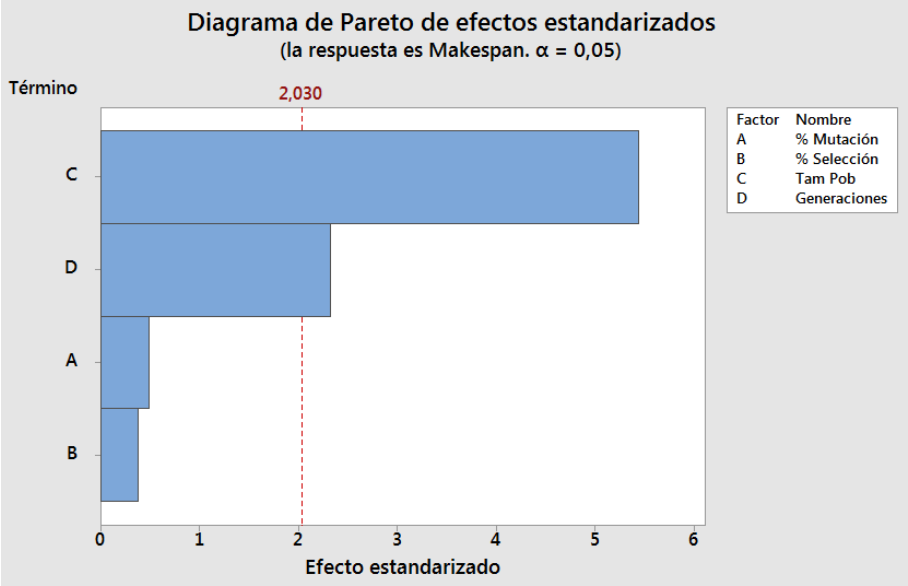
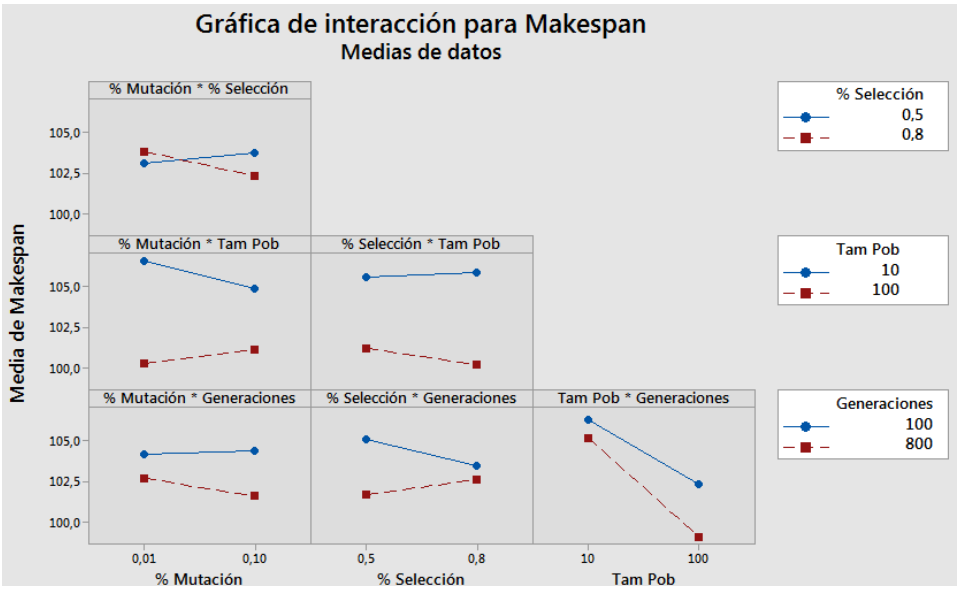


Tabla 8. Análisis de varianza para el problema EX31.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	2,025	2,025	0,24	0,630
% Selección	1	1,225	1,225	0,14	0,708
Tam Pob	1	255,025	255,025	29,71	0,0001
Generaciones	1	46,225	46,225	5,38	0,026
Error	35	300,475	8,585		
Total	39	604,975			

Para esta instancia, se puede observar que el tamaño de la población y el número de generaciones son los factores significativos para obtener un mejor valor de *makespan*. De la gráfica a de efectos podemos concluir que su relación es inversamente proporcional, es decir, a mayores niveles de tamaño de población y número de generaciones se obtendrá un menor *makespan*.

Figura 30. Gráfica de interacción para makespan para el problema EX31.



Adicionalmente, como se puede observar en la figura 30, no existe interacción significativa entre los factores, y para el caso en el que si existe interacción, estos no son significativos en cuanto a la variable respuesta.

9.5.5. Análisis de varianzas para el problema EX32

Figura 31. Diagrama de Pareto de efectos estandarizados del problema EX32.

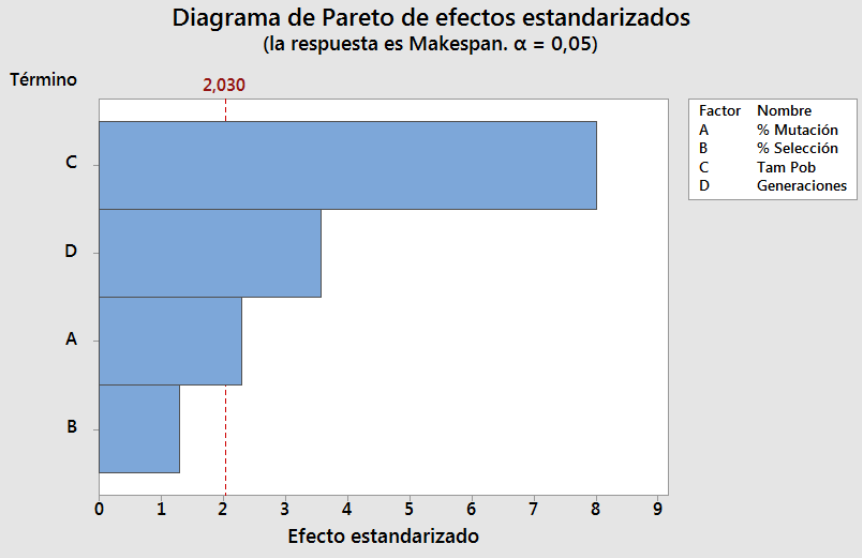


Tabla 9. Análisis de varianzas para el problema EX32.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	25,6	25,6	5,23	0,028
% Selección	1	8,1	8,1	1,65	0,207
Tam Pob	1	313,6	313,6	64,07	0,0001
Generaciones	1	62,5	62,5	12,77	0,001
Error	35	171,3	4,894		
Total	39	581,1			

En este diseño se observó que adicional a lo observado en los diseños anteriores (el tamaño de población y número de generaciones son significativos), la mutación se sumó a estos dos factores, obteniéndose un tercer factor relevante para obtener un menor valor de variable respuesta. Para los dos primeros factores, un nivel alto de los factores generará un mejor valor de la variable respuesta. En conclusión,

para obtener una mejor solución, se debe utilizar la siguiente combinación de niveles de los factores: C^+ D^+ y A^-.

9.5.6. Análisis de varianzas para el problema EX43

Figura 32. Diagrama de Pareto de efectos estandarizados del problema EX43.

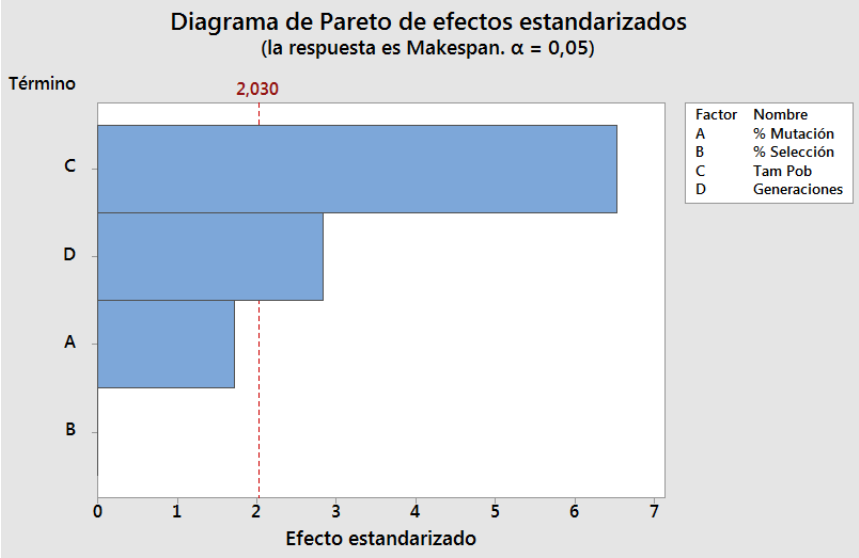


Tabla 10. Análisis de varianzas para el problema EX43.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	19,6	19,6	2,97	0,093
% Selección	1	0	0	0	1
Tam Pob	1	280,9	280,9	42,63	0,0001
Generaciones	1	52,9	52,9	8,03	0,008
Error	35	230,6	6,589		
Total	39	584			

En la tabla 10 se observa que en para la solución de esta instancia, el tamaño de la población y el número de generaciones en su nivel alto (100 y 800 respectivamente) contribuyen a obtener un mejor valor de *makespan*. Adicionalmente, se observa que no hay relevancia en el porcentaje de selección para obtener una mejora en la variable respuesta.

9.5.7. Análisis de varianzas para el problema EX62

Figura 33. Diagrama de Pareto de efectos estandarizados del problema EX43.

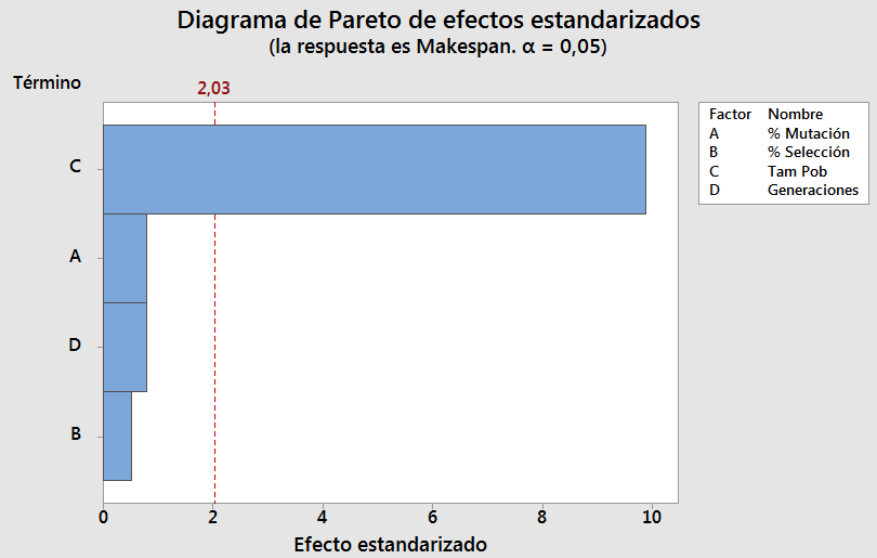


Tabla 11. Análisis de varianzas para el problema EX62

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	0,9	0,9	0,61	0,441
% Selección	1	0,4	0,4	0,27	0,606
Tam Pob	1	144,4	144,4	97,57	0,0001
Generaciones	1	0,9	0,9	0,61	0,441
Error	35	51,8	1,48		
Total	39	198,4			

Para la instancia EX82, se observa que el tamaño de la población (C) es significativo en la variable de respuesta *makespan*, puesto que su valor p es inferior a 0,05. Los demás efectos principales y las interacciones no generan inciden significativa en la respuesta del problema.

9.5.8. Análisis de varianzas para el problema EX82

Figura 34. Diagrama de Pareto de efectos estandarizados del problema EX43.

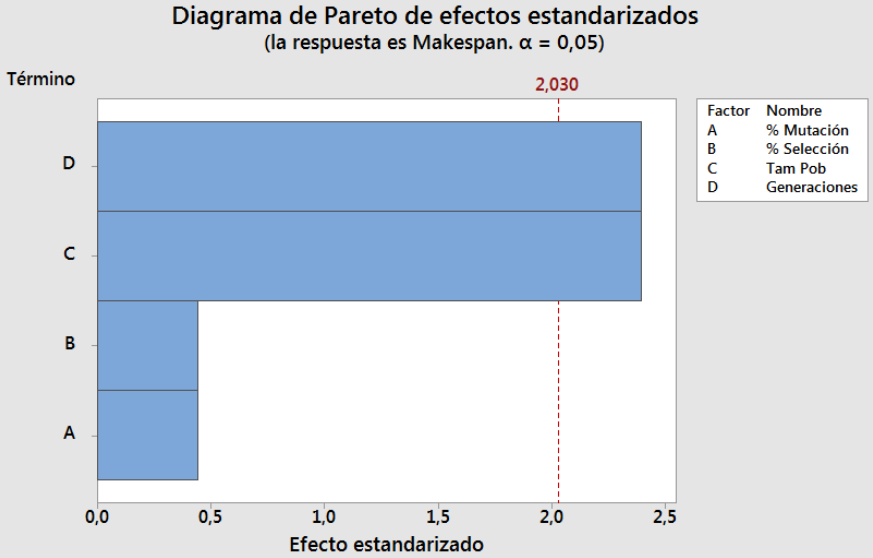


Tabla 12. Análisis de varianzas para el problema EX82.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	2,025	2,025	0,19	0,663
% Selección	1	2,025	2,025	0,19	0,663
Tam. Pob	1	60,025	60,025	5,73	0,022
Generaciones	1	60,025	60,025	5,73	0,022
Error	35	366,875	10,482	-	-
Total	39	490,975	-	-	-

En la tabla 12 se observa que para este problema, son 2 los efectos principales significativo en la variable respuesta, puesto que presentan un valor P menor a 0,05. El tamaño de la población y el número de generaciones, en igual magnitud. Los mejores resultados de variable respuesta se obtendrán con un nivel alto de estos dos factores. Ni las interacciones ni los otros dos factores son ejercen algún tipo de acción significativa sobre el *makespan*.

9.5.9. Análisis de varianzas para el problema EX93

Figura 35. Diagrama de Pareto de efectos estandarizados del problema EX93.

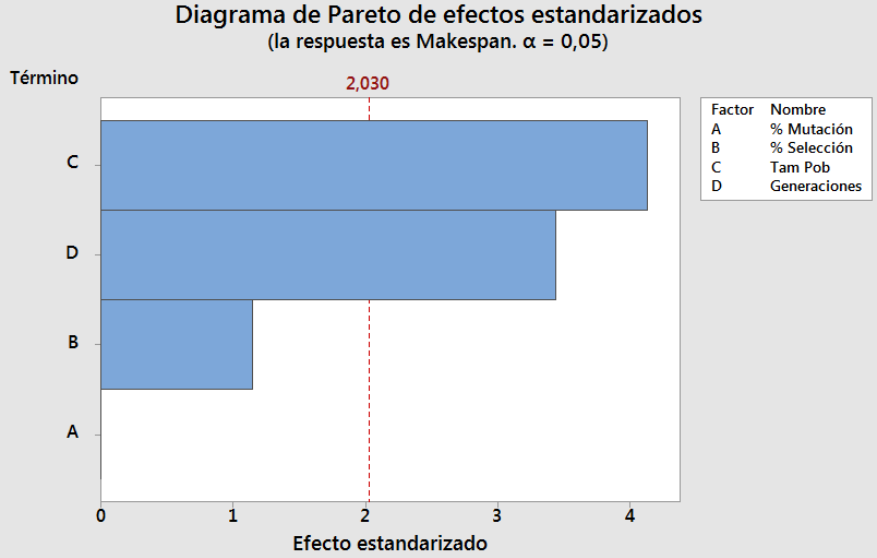


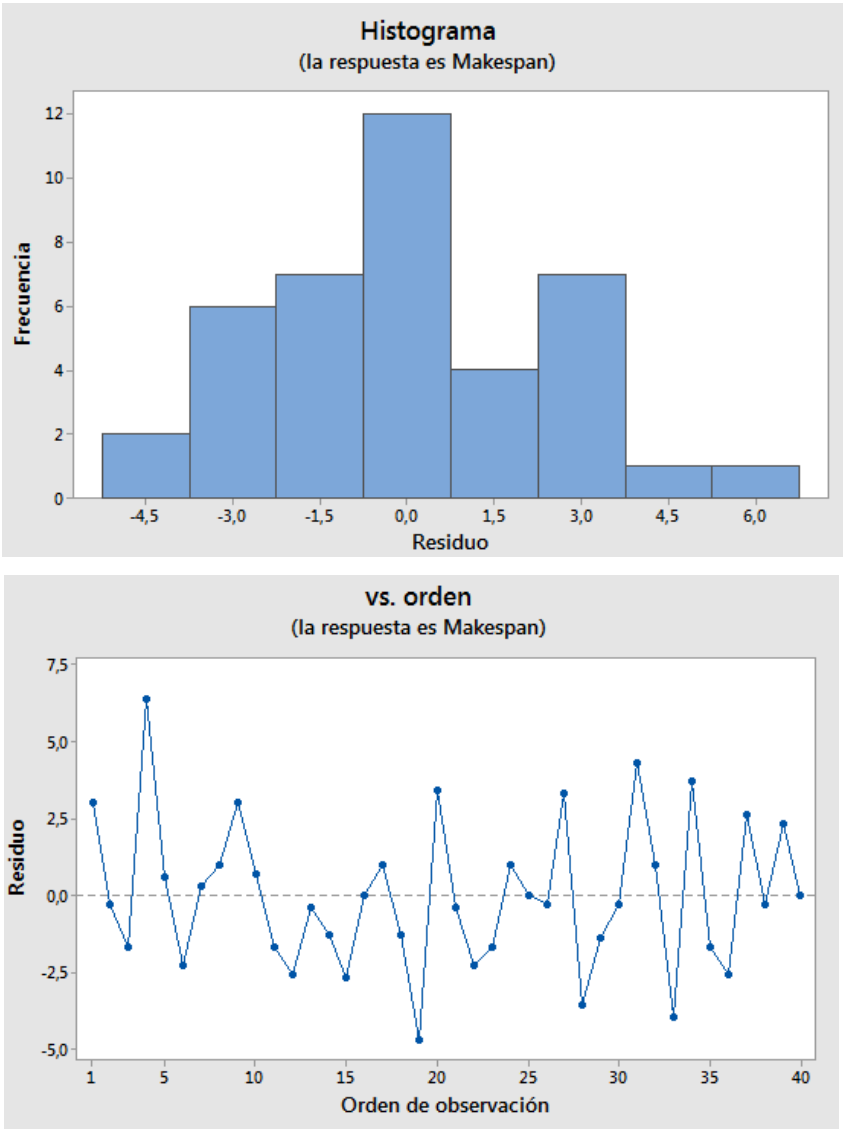
Tabla 13. Análisis de varianzas para el problema EX93.

Análisis de varianza					
Fuente	**GL**	**SC Ajust**	**MC Ajust**	**Valor f**	**Valor p**
% Mutación	1	0	0	0	1
% Selección	1	2,5	2,5	1,32	0,259
Tam Pob	1	32,4	32,4	17,05	0,0001
Generaciones	1	22,5	22,5	11,84	0,002
Error	35	66,5	1,9		
Total	39	123,9			

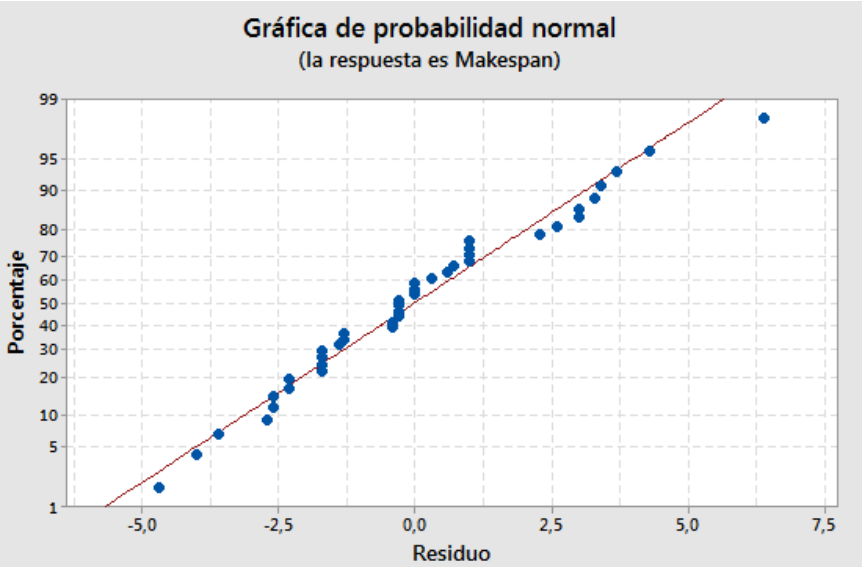
Finalmente, para la instancia 7 dentro de las validadas, se observa que nuevamente son el tamaño de población y el número de generaciones los que representan significancia sobre la variable respuesta. Los niveles altos de estos dos factores son los que inciden para obtener un menor valor de *makespan*. El porcentaje de mutación en este problema no tiene relevancia alguna, tal y como se observa en la Figura 35.

Para todos los diseños factoriales realizados anteriormente se validaron los distintos supuestos de normalidad, varianzas constante e independencia, a través de la información gráfica obtenida por medio del software estadístico utilizado (Minitab 17).[106]

Figura 36. Gráfica de validación de supuestos del análisis de varianza para la instancia 4.



[106] Por medio de la gráfica de residuales versus orden de observación, histograma de residuales para verificar la tendencia normal de los datos, y la gráfica de probabilidad normal.



De lo obtenido a través de los distinto diseños factoriales elaborados, se puede concluir que aproximadamente el 87% de los casos el tamaño de la población es significativo para poder obtener mejores resultados de la variable respuesta. Esto se puede interpretar, junto con el 62% en el cual es representativo el número de generación, que el algoritmo es evolutivo, lo que evidentemente hace que dichos factores deban ser analizados para mejores soluciones.

10. CONCLUSIONES

De acuerdo a lo observado a lo largo de la investigación y la validación hecha en el ANEXO B se observó la relevancia del tipo de codificación a utilizar en el algoritmo genético en problemas de mayor complejidad.

Como se observó en el diseño de experimentos, más del 70% de los casos analizados demuestran la significancia del número de generaciones, lo cual podría tomarse como un resultado esperado al tener en cuenta que se está trabajando con un algoritmo evolutivo.

Igualmente, se determinó que en el 100% de los casos el tamaño de población fue un factor significativo en función de la minimización del *makespan*. Esto, hace ver que aun cuando los factores de probabilidad de cruce y mutación no fueron representativos en la mayoría de casos, es necesario contar con una cantidad amplia de cromosomas para que operen estos elementos del algoritmo y obtener mejores resultados a medida que se avance en el número de iteraciones. Contar con un tamaño de población elevado, puede verse reflejado en una mayor exploración del espacio de solución.

Durante la validación de factores se observó que un valor elevado en el porcentaje de mutación hace que se pierda la información genética y se empiece a realizar, en vez de un proceso evolutivo de búsqueda de mejores soluciones, una búsqueda aleatoria.

De lo obtenido durante la investigación, se obtuvo que al realizar un procedimiento de cruce de cromosomas de más de 3 puntos, la información genética de los cromosomas padres puede perderse por la cantidad de particiones realizadas a los mismos. Finalmente, de los resultados obtenidos en las distintas instancias, se puede concluir que el pseudocódigo generado de algoritmo genético es una

metaheurística efectiva a la hora de abarcar problemas de programación de operaciones con limitaciones de transporte.

11. RECOMENDACIONES

Debido al margen de error manejado en las instancias donde el transporte se convierten en operaciones complejas de secuenciar, se hace necesario el uso de alguna metodología hibrida o una heurística que opere dentro del problema de asignación y secuenciación de operaciones y recursos relacionados con dicha actividad.

Se recomienda para posteriores investigaciones, realizar inclusiones a este tipo de problemas con características más cercanas a lo que se presenta en el mundo empresarial (Tiempos de set up, ventana de tiempo de procesamiento, entre otros), de manera que se pueda presentar una aplicación de esta soluciones a problemas cotidianos.

De acuerdo a lo observado durante la ejecución del algoritmo, se recomienda generar una alianza entre el parque tecnológico Guatiguará y el grupo de investigación OPALO, para poder hacer uso de unidad de supercomputación para poder realizar ejecuciones de distintos algoritmos y disminuir los tiempos computacionales.

Adicionalmente, se recomienda profundizar en la mejora del algoritmo, de manera que se pueda optimizar el tiempo de cómputo y la forma de asignación y secuenciamiento de las distintas operaciones.

BIBLIOGRAFÍA

ABDELMAGUID, Tamer F., et al. A hybrid GA/heuristic approach to the simultaneous scheduling of machines and automated guided vehicles. En: International journal of production research, 2004, vol. 42, no 2, p. 267-281.

AGUILAR, Karin y PÉREZ, Yuleiny. Un algoritmo memético para la minimización del makespan en el problema del Job shop scheduling. Bucaramanga, 2012, 223p. Tesis (ingeniería industrial). Universidad industrial de Santander. Facultad de ingeniería físicomecánicas. Escuela de estudios industriales y empresariales.

BAGHERI, A., et al. An artificial immune algorithm for the flexible job-shop scheduling problem. En: Future Generation Computer Systems, 2010, vol. 26, no 4, p. 533-541.

BAKER, James Edward. Adaptive selection methods for genetic algorithms. En Proceedings of an International Conference on Genetic Algorithms and their applications. 1985. p. 101-111.

BAKER, Kenneth Robert. Introduction to sequencing and scheduling. New York: Wiley, 1974.

BŁAŻEWICZ, Jacek; DOMSCHKE, Wolfgang; PESCH, Erwin. The job shop scheduling problem: Conventional and new solution techniques. En: European journal of operational research, 1996, vol. 93, no 1, p. 1-33.

BLICKLE, Tobias; THIELE, Lothar. A Mathematical Analysis of Tournament Selection. En ICGA. 1995. p. 9-16.

BRANDIMARTE, Paolo. Routing and scheduling in a flexible job shop by tabu search. En: Annals of Operations research, 1993, vol. 41, no 3, p. 157-183.

BRUCKER, Peter. Scheduling algorithms. Fifth edition, Berlin, Springer, 2007, 367p.

BRUCKER, Peter; BURKE, Edmund K.; GROENEMEYER, Sven. A branch and bound algorithm for the cyclic job-shop problem with transportation. En: Computers & Operations Research, 2012, vol. 39, no 12, p. 3200-3214.

CONWAY, Richard W.; MAXWELL, William L.; MILLER, Louis W. Theory of scheduling. Courier Dover Publications, 1967, 294p.

CASTRILLÓN, Omar D.; SARACHE, William A.; GIRALDO, Jaime A. Aplicación de un algoritmo evolutivo en la solución de problemas Job Shop-Open Shop. En: Información tecnológica, 2011, vol. 22, no 1, p. 83-92.

CASTRILLON, Omar; SARACHE, William; GIRALDO, Jaime. Job Shop methodology based on an Ant Colony. En: Dyna, 2009, vol. 76, no 159, p. 177-184.

CORREA, Alexander A.; RODRIGUEZ, Elkin; LONDOÑO María I., Secuenciación de operaciones para configuraciones de planta tipo flexible Job Shop: Estado del arte. En: Avances en Sistemas e Informática. Diciembre, 2008. vol. 5, no. 3, p. 151-161.

DEROUSSI, L.; GOURGAND, M.; TCHERNEV, N. A simple metaheuristic approach to the simultaneous scheduling of machines and automated guided vehicles. En: International Journal of Production Research, 2008, vol. 46, no 8, p. 2143-2164.

DEROUSSI, L.; NORRE, S. Simultaneous scheduling of machines and vehicles for the flexible job shop problem. En: International conference on metaheuristics and nature inspired computing. 2010.

DE GIOVANNI, L.; PEZZELLA, Ferdinando. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. En: European journal of operational research, 2010, vol. 200, no 2, p. 395-408.

DUARTE, Abraham; PANTRIGO, Juan; GALLEGO Micael. Introducción a la optimización. En: Metaheuristicas. Madrid: Dykinson, 2008. p. 1-14.

FATTAHI, Parviz; MEHRABAD, Mohammad Saidi; JOLAI, Fariborz. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. En: Journal of Intelligent Manufacturing, 2007, vol. 18, no 3, p. 331-342.

FERNÁNDEZ, A. D., Velarde, J. G., Laguna, M., Mascato, P., Tseng, F. T., Glover, F., y Ghaziri, H. M. Optimización heurística y redes neuronales. Madrid: Paraninfo SA, 1996.

GAO, Jie; GEN, Mitsuo; SUN, Linyan. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. En: Journal of Intelligent Manufacturing, 2006, vol. 17, no 4, p. 493-507.

GAREY, Michael y JOHNSON, David. Computer and intractability: A guide to theory of NP-Completeness. 1979

GAREY, Michael R.; JOHNSON, David S.; SETHI, Ravi. The complexity of flowshop and jobshop scheduling. En: Mathematics of operations research, 1976, vol. 1, no 2, p. 117-129.

GONZÁLEZ, Miguel A.; VELA, Camino R.; VARELA, Ramiro. An Efficient Memetic Algorithm for the Flexible Job Shop with Setup Times. En: ICAPS. 2013

GONZALEZ, Miguel A. Soluciones Metaheuristicas al “Job-Shop Scheduling Problem with Sequence-Dependent Setup Times”. Tesis Doctoral. España: Universidad de Oviedo. Departamento de informática. Julio 2011. 2810 p.

GRAHAM, Ronald L., et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. En: Annals of discrete Mathematics, 1979, vol. 5, p. 287-326.

GOLDBERG, David E. Genetic Algorithms in Search, Optimization and Machine Learning. En: Addison-Weasley Longman Publishing Co. 1989. p. 372. ISBN:0201157675.

HERNÁNDEZ, José Luis. Algoritmos genéticos y su aplicación en optimización de redes. 1998. Tesis Doctoral. Facultad de Informática.

HOLLAND, John H. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. En: The University of Michigan Press, 1975. ISBN: 0262580989.

HURINK, Johann; KNUST, Sigrid. Makespan minimization for flow-shop problems with transportation times and a single robot. En: Discrete Applied Mathematics, 2001, vol. 112, no 1, p. 199-216.

HURINK, Johann; KNUST, Sigrid. Tabu search algorithms for job-shop problems with a single transport robot. En: European journal of operational research, 2005, vol. 162, no 1, p. 99-111.

JAIN, Anant Singh; MEERAN, Sheik. Deterministic job-shop scheduling: Past, present and future. En: European journal of operational research, 1999, vol. 113, no 2, p. 390-434.

JANSEN, Klaus; MASTROLILLI, Monaldo; SOLIS-OBA, Roberto. Approximation algorithms for flexible job shop problems. En: LATIN 2000: Theoretical Informatics. Springer Berlin Heidelberg, 2000. p. 68-77.

JOHNSON, Selmer Martin. Optimal two-and three-stage production schedules with setup times included. En: Naval research logistics quarterly, 1954, vol. 1, no 1, p. 61-68.

KIRKPATRICK, Scott; GELATT, Daniel y VECCHI, Mario P. Optimization by simulated annealing. En: Science, 1983. p. 671-680.

KOKASH, Natalia. An introduction to heuristic algorithms. Departamento de Informática y Telecomunicaciones. 2005.

LACOMME, Philippe; LARABI, Mohand; TCHERNEV, Nikolay. Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles. En: International Journal of Production Economics, 2013, vol. 143, no 1, p. 24-34.

LAWLER, Eugene L., et al. Sequencing and scheduling: Algorithms and complexity. En: Handbooks in operations research and management science, 1993, vol. 4, p. 445-522.

LIU, Zhuangcheng, et al. Solving multi-objective Flexible Job Shop Scheduling with transportation constraints using a micro artificial bee colony algorithm. En: Computer Supported Cooperative Work in Design (CSCWD), 2013 IEEE 17th International Conference on. IEEE, 2013. p. 427-432.

LÓPEZ DÍAZ, José Carlos. Un algoritmo genético con codificación real para la evolución de transformaciones lineales. 2010.

MICHALEWICZ, Zbigniew. Genetic algorithms+ data structures= evolution programs. Springer, 1996.

MARTÍ, Rafael. Procedimientos Metaheurísticos en optimización combinatoria. [En línea] Universidad de Valencia, Departamento de Estadística e investigación operativa. (2003). P. 1-60. [Consultado 28 ene. 2013]. Disponible en: <http://www.uv.es/~rmartí/paper/cpapers.html>

MEDINA DURÁN, Rosa; PRADENAS ROJAS, Lorena; PARADA DAZA, Víctor. Un algoritmo genético para el problema de Job Shop Flexible. En: Ingeniare. Revista chilena de ingeniería, 2011, vol. 19, no 1, p. 53-61.

MILLER, Brad L.; GOLDBERG, David E. Genetic algorithms, tournament selection, and the effects of noise. Complex Systems, 1995, vol. 9, no 3, p. 193-212.

MOSLEHI, Ghasem; MAHNAM, Mehdi. A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. En: International Journal of Production Economics, 2011, vol. 129, no 1, p. 14-22.

MONTREUIL, B. Fractal layout organization for job shop. En: International Journal of Production Research. 1999, vol. 37, no. 3, 501-521

MUÑOZ, Mario A. Estrategias Evolutivas aplicadas a los Algoritmos de Enjambres para el control de sistemas complejos. Tesis de Maestría. Santiago de Cali: Universidad del Valle. Facultad de Ingeniería y Electrónica. 2008. 90 p.

NOWICKI, Eugeniuz y SMUTNICKI, Czeslaw. A fast taboo search algorithm for the job shop problem. En: Management science. Junio 1996. vol 42. P. 797-813.

OSMAN, I.H. y Kelly, J.P. Meta-Heuristics: Theory and Applications. Boston: Kluwer Academic, 1996.

PALACIOS, Melissa; JAIMES, Christian. Alternativa de solución al problema de distribución de planta para instalaciones de áreas iguales y desiguales mediante un Algoritmo Hibrido Genético. Proyecto de grado. Bucaramanga: Universidad Industrial de Santander. Facultad de Ingeniería Físico-Mecánicas. 2011. 139 p.

RADCLIFFE, Nicholas J. Equivalence class analysis of genetic algorithms. En: Complex systems, 1991, vol. 5, no 2, p. 183-205.

REEVES, Colin R. Modern heuristic techniques for combinatorial problems. En: John Wiley & Sons Inc., 1993.

ROSSI, A. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. En: International journal of production economics, 2014, vol. 153, no 1, p. 253-267.

SAIDI-MEHRABADA, Mohammad y FATTAHI, Parviz. Flexible job shop scheduling with tabu search algorithms. En: The Internal Journal Of Advanced Manufacturing Technology. Mayo, 2006. vol. 32. p. 563-570.

SANKAR, S. Saravana; PONNANBALAM, S. G.; RAJENDRAN, C. A multiobjective genetic algorithm for scheduling a flexible manufacturing system. En: The International Journal of Advanced Manufacturing Technology, 2003, vol. 22, no 3-4, p. 229-236.

SARIÇIÇEK, İnci y CENK, Çelik. Two meta-heuristics for parallel machine scheduling with job splitting to minimize total tardiness. En: Applied Mathematical Modelling. 2011. vol. 35. p. 4117-4126.

SARMIENTO, Cindy Johanna. Metodo particle swarm optimization (PSO- Enjambre de partículas) aplicado al problema de múltiples objetivos del Job Shop Scheduling (JSP) o secuenciamiento de máquinas. Bucaramanga, 2012, 134p. Tesis (ingeniería industrial). Universidad industrial de Santander. Facultad de ingeniería físicomecánicas. Escuela de estudios industriales y empresariales.

TANG, Jianchao, et al. A hybrid algorithm for flexible job-shop scheduling problem. En: Procedia Engineering, 2011, vol. 15, p. 3678-3683.

VIDAL, Aitana. Algoritmos heurísticos en optimización. Tesis de Maestría. Santiago de Compostela: Universidad de Santiago de Compostela. Facultad de Matemáticas. 2013. 94 p.

XIA, Weijun; WU, Zhiming. An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. En: Computers & Industrial Engineering, 2005, vol. 48, no 2, p. 409-425.

ZANDIEH, M.; MAHDAVI, I y BAGHERI, A. Solving the Flexible job-Shop Scheduling Problem by a Genetic Algorithm. En: Journal of Applied Sciences. 2008. ISBN 1812-5654.

ZHANG, Qiao; MANIER, Hervé; MANIER, M.-A. A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times. En: Computers & Operations Research, 2012, vol. 39, no 7, p. 1713-1723.

ZHANG, Guohui, et al. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. En: Computers & Industrial Engineering, 2009, vol. 56, no 4, p. 1309-1318.

ANEXOS

ANEXO A. VERIFICACIÓN IMPORTANCIA DE LA CODIFICACIÓN EN EL ALGORITMO GENÉTICO

Durante la etapa inicial de la construcción del algoritmo, se pudo observar que distintos autores concluyen de sus investigaciones que la codificación en el Algoritmo genético se convierte en una de las etapas críticas para el funcionamiento óptimo del algoritmo. Para ello, se realizó la validación del mismo, partiendo de las 3 posibles codificaciones a utilizar para abordar el tema de investigación.

1. Codificación por *Jobs*.

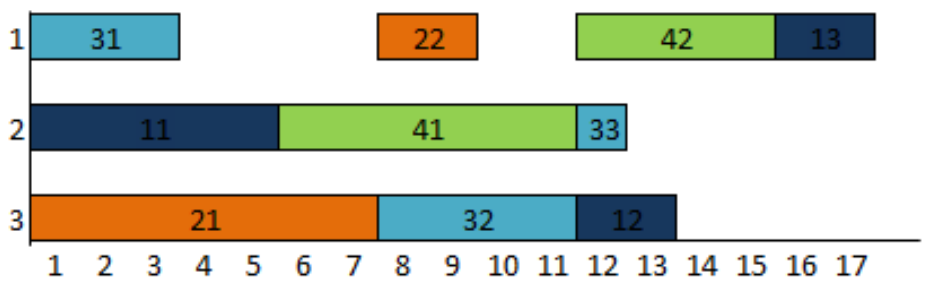
En este tipo de codificación, lo que se indica es que el cromosoma contendrá tantos bits como trabajos existan en el problema. A partir de allí, el algoritmo realizará la asignación de operaciones de acuerdo a la restricción de precedencia que existe junto con ellas, realizándolo de acuerdo a la posición que ocupe el trabajo dentro del cromosoma.

Ejemplo: Para un problema donde se cuenta con 4 trabajos, cada uno con hasta 3 operaciones, se realizaría la construcción del Gantt de la siguiente manera:

[3 1 2 4]

Gantt asignando la primer operación del job 3, gantt asignando la primer operación del job 1,gantt asignando la primer operación del job 2.

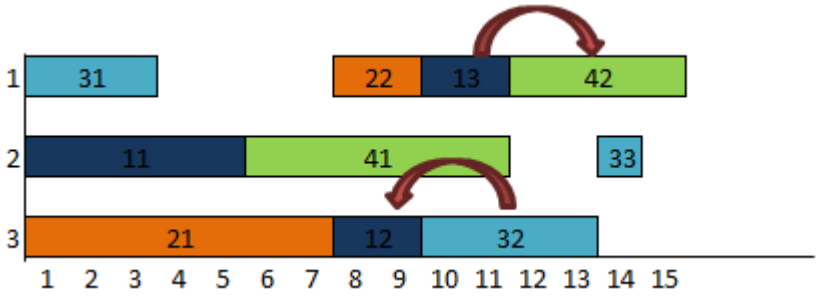
Figura 37. Representación gráfica de la asignación por trabajos.



2. Codificación por operaciones.

Para esta codificación, se pueden presentar dos tipos de situaciones: En primer lugar, obtener un cromosoma desde 1 hasta J*O números (Donde J es el numero de *Jobs* y O el número de operaciones), donde cada número sea asignado a una operación de determinado trabajo. Y en segundo lugar, un cromosoma conformado por el número de *Jobs* tantas veces como operaciones hayan.

Figura 38. Representación gráfica de la asignación por operaciones.



Se decidió realizar un tipo de codificación descrita por el número de *Jobs* repetitivos, puesto que se indica que, con base en diferentes artículos estudiados, podrían obtenerse individuos infactibles en el proceso de mutación o cruce, o simplemente limitar al uso de algunos de los distinto tipos de estas etapas, todo esto con el fin de no ocupar mayor espacio de máquina al obtener procesos repetitivos de conversión de individuos infactibles a factibles, o mayor complejidad en el manejo del problema.

Como se observa en las figuras anteriores, al permitir la modificación en el intercambio de operaciones de acuerdo a la codificación utilizada, se obtendrá un mejor valor de solución a partir de dicha codificación. Todo esto, partiendo del hecho de que por medio de esta codificación, el espacio de solución será mayormente explorado en comparación a la codificación de cromosoma por Jobs.

Para finalizar de validar, se decidió correr unas instancias previas con estos tipos de codificaciones, observando la variación de la solución en cada caso. Se tomaron

las configuraciones de Jobs de las instancias propuestas por Deroussi[107] sin tener en cuenta e transporte, para realizar un test que permita validar.

Tabla 14. Comparación de la programación por trabajos y por operaciones

Cruce:0.8 Mutación: 0.1 Gener:300 Pob:100

Instancia	Total de Operaciones	C_{MAX} Codificación por operaciones	C_{MAX} Codificación por trabajos	% Error
Jobset 1	13	60	60	0%
Jobset 2	15	82	82	0%
Jobset 3	16	70	77	10%
Jobset 4	20	52	56	7,69%
Jobset 5	13	48	48	0%
Jobset 6	18	89	94	5,62%
Jobset 7	19	66	68	3,03%
Jobset 8	20	141	145	2,84%
Jobset 9	18	81	87	7,41%

Como se puede observar, la codificación puede llegar a presentar una disminución en el *makespan* de hasta 10% según la complejidad del problema (en termino de numero de *jobs*, operaciones y máquinas).

[107] Deroussi, L. Op cit.

ANEXO B. INSTANCIAS MODIFICADAS DE DEROUSSI PARA VALIDAR EL ALGORITMO DE TIPO JOB SHOP FLEXIBLE.

Las instancias testeadas dentro del capítulo 7.2 fueron establecidas a partir de las propuestas por Deroussi et al[108], a las cuales se les realizó la modificación de configuración del *Jobset* de manera aleatoria, para poder obtener de estas mismas instancias una configuración de Job shop flexible.

A continuación se encuentran las instancias modificadas, donde la matriz inicial describe los tiempos de ejecución de cada una de las operaciones; y posteriormente, se observa la matriz de máquinas habilitadas para realizar cada una de ellas.

- *Jobset 1*

	J1	J2	J3	J4	J5
O1	8	20	12	14	10
O2	16	10	8	18	15
O3	12	18	15	0	0

	J1	J2	J3	J4	J5
O1	m1	m1	m3	m4	m3
O2	m2	m2-m3	m4	m2	m1
O3	m4	m2	m1	m2-m4	m3

- *Jobset 2*

	J1	J2	J3	J4	J5	J6
O1	10	10	10	10	10	10
O2	18	18	20	15	15	15
O3	0	0	0	12	12	12

	J1	J2	J3	J4	J5	J6
O1	m1	m2	m1	m2-m3-m4	m1-m4	m1
O2	m4	m4	m1-m3	m3-m4	m2	m2
O3	m4	m2	m1	m3-m4	m4	m1-m2-m3

[108] Ibíd.

- *Jobset 3*

	J1	J2	J3	J4	J5	J6
O1	16	18	20	15	8	10
O2	15	15	10	10	10	15
O3	0	0	0	0	15	8
O4	0	0	0	0	17	15

	J1	J2	J3	J4	J5	J6
O1	m1	m2-m4	m1	m3	m1-m4	m2
O2	m3	m4	m2	m1-m4	m2	m3
O3	m4	m2	m1	m4	m3	m4
O4	m4	m2	m1	m3-m4	m4	m1-m2-m3

- *Jobset 4*

	J1	J2	J3	J4	J5
O1	11	12	7	7	9
O2	10	10	10	8	7
O3	7	8	9	12	8
O4	0	0	8	6	10

	J1	J2	J3	J4	J5
O1	m1-m4	m3	m2	m2-m4	m1
O2	m1-m3	m2	m3	m4	m2-m3
O3	m2	m2-m4	m1	m1-m3	m4
O4	m4	m1-m4	m4	m2-m4	m2-m3

- *Jobset 5*

	J1	J2	J3	J4	J5
O1	6	18	9	6	3
O2	12	6	3	15	9
O3	9	15	12	0	0

	J1	J2	J3	J4	J5
O1	m1	m1-m2	m3	m4	m3
O2	m2	m3	m4	m2	m1-m4
O3	m4	m2	m1-m4	m4	m4

- *Jobset 6*

	J1	J2	J3	J4	J5	J6
O1	9	19	14	14	11	10
O2	11	20	20	20	16	12
O3	7	13	9	9	8	10

	J1	J2	J3	J4	J5	J6
O1	m1	m1-m4	m2	m2-m4	m1	m1
O2	m2	m2	m3	m3-m4	m1-m3	m3
O3	m4	m4	m1-m4	m4	m2-m4	m3-m4

- *Jobset 7*

	J1	J2	J3	J4	J5	J6	J7	J8
O1	6	11	9	16	9	13	10	11
O2	6	9	7	7	18	19	9	9
O3	0	0	0	0	0	6	13	8

	J1	J2	J3	J4	J5	J6
O1	m1	m1-m4	m2	m2-m4	m1	m1
O2	m2	m2	m3	m3-m4	m1-m3	m3
O3	m4	m4	m1-m4	m4	m2-m4	m3-m4

- *Jobset 8*

	J1	J2	J3	J4	J5	J6
O1	12	12	12	12	10	10
O2	21	21	21	21	14	14
O3	11	11	11	11	18	18
O4	0	0	0	0	9	9

	J1	J2	J3	J4	J5	J6	J7	J8
O1	m1	m2	m2	m3-m4	m1	m2-m4	m1-m3	m1
O2	m4	m4	m1-m4	m4	m1-m3	m3	m2	m2
O3	m4	m4	m4	m4	m4	m4	m2-m3	m4

- *Jobset 9*

	J1	J2	J3	J4	J5
O1	9	16	21	20	14
O2	12	11	18	22	16
O3	9	9	7	11	13
O4	6	0	0	0	9

	J1	J2	J3	J4	J5	J6
O1	m2	m2-m4	m2-m3	m2	m1-m3	m1
O2	m1-m3	m3	m3	m3	m2	m2-m4
O3	m2-m4	m4	m4	m4	m2-m3	m3
O4	m4	m4	m4	m4	m3-m4	m1-m2-m4