HERRAMIENTA SOFTWARE PARA EL METODO PSO (PARTICLE SWARM OPTIMIZATION) APLICADO AL PROBLEMA DE MULTIPLES OBJETIVOS DEL JSP (JOB SHOP PROBLEM)

CARLOS EDINSON PINZÓN LEON

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECANICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2013

HERRAMIENTA SOFTWARE PARA EL METODO PSO (PARTICLE SWARM OPTIMIZATION) APLICADO AL PROBLEMA DE MULTIPLES OBJETIVOS DEL JSP (JOB SHOP PROBLEM)

CARLOS EDINSON PINZÓN LEON

Trabajo de Grado presentado como
requisito para optar al título de
Ingeniero de Sistemas

Director
MSc. FERNANDO ROJAS MORALES

Codirector
Ph.D. HENRY LAMOS DIAZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECANICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMATICA
BUCARAMANGA
2013

AGRADECIMIENTOS

A la **UNIVERSIDAD INDUSTRIAL DE SANTANDER** y a la **ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA** por la formación brindada.

Al **MAESTRO FERNANDO ROJAS**, DIRECTOR DE PROYECTO, quién compartió conmigo acerca de lo más importante, **DIOS**.

Al **DOCTOR HENRY LAMOS**, CODIRECTOR DE PROYECTO, por brindarme la oportunidad de desarrollar esta investigación, por su paciencia y por compartir conmigo su experiencia y conocimiento.

A **MARTHA PRADA**, por su apoyo incondicional y sincero, por compartir tanto conmigo.

A mi **MADRE y ABUELA** por confiar en que el momento llegaría.

A los demás.

DEDICATORIA

A DIOS Y LOS QUE QUIERO.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

RESUMEN

TÍTULO: HERRAMIENTA SOFTWARE PARA EL METODO PSO (PARTICLE SWARM OPTIMIZATION) APLICADO AL PROBLEMA DE MULTIPLES OBJETIVOS DEL JSP (JOB SHOP PROBLEM)*

AUTOR: Carlos Edinson Pinzón Leon**

PALABRAS CLAVES: Programación de tareas, metaheurística, enjambre de partículas, optimización combinatoria, job shop, programación multi-objetivo, software.

DESCRIPCIÓN:

En una investigación previa (SARMIENTO, 2012) se desarrolló un pseudocódigo para resolver el problema de secuenciamiento de máquinas en empresas tipo taller (Job Shop Scheduling) con múltiples objetivos. Este pseudocódigo fue basado en el método Multi-objective Particle Swarm Optimization (MOPSO), variante del PSO original diseñado para un solo objetivo.

La metaheurística MOPSO permite aproximar soluciones de problemas combinatorios de tipo NP-Hard en tiempos computacionales razonables. En este trabajo se presenta la herramienta software desarrollada en lenguaje JAVA, implementando el pseudocódigo planteado anteriormente y considerando como objetivos a minimizar el tiempo máximo de finalización de la última tarea (makespan) y la tardanza máxima (maximum lateness).

Finalmente, la herramienta es probada para una instancia de 5X5 del problema JSP, los resultados muestran que la aplicación desarrollada logra aproximar al conjunto solución Frente de Pareto. Cabe resaltar que a causa de la explosión factorial que sufre el problema de JSP en la medida que el tamaño de las instancias crecen en los parámetros de entrada número de trabajos y de máquinas, la aplicación software aumenta el tiempo de respuesta para el cálculo de soluciones, dejando en evidencia la necesidad de abordar este tipo de problemas en arquitecturas computacionales tipo clúster.

---

* Proyecto de Grado. Modalidad: Trabajo de investigación.

** Facultad de Ingenierías Físico - Mecánicas. Escuela de Ingeniería de Sistemas e Informática. Director: MSc. Fernando Antonio Rojas Morales. Co-Director: Ph.D. Henry Lamos Díaz.

ABSTRACT

TITLE: SOFTWARE FOR THE METHOD PSO (PARTICLE SWARM OPTIMIZATION) APPLIED TO THE PROBLEM OF MULTIPLE OBJECTIVES JSP (JOB SHOP PROBLEM)[*]

AUTHOR: Carlos Edinson Pinzón León[**]

KEYWORDS: Scheduling**,** metaheuristic, particle swarm, combinatorial optimization, job shop, multi-objective programming, software.

DESCRIPTION:

This work was based on a research which developed a pseudocode to solve the problems of sequence of machines in the field of production with multiple objectives. The method Multi-objective Particle Swarm Optimization (MOPSO) was the root for the deployment of this pseudocodeit is a variant of the original PSO (one objective).

MOPSO allows approaching the solutions to type NP-Hard combinatorial problems for the resolution of problems by means of the execution of an algorithm. The working tool was developed in a language JAVA by using the pseudocode MOPSO. Also, was considered the optimization of time used on the last task, more known as makespan and maximum lateness.

Finally, this tool was tried for an authority of 5X5 of the problem JSP. The results demonstrated that the developed application approaches the solution given by Pareto Font. An important point to have in mind was the combinatorial explosion of the JSP compared with the sample. That is when the inputs of works and machines increase, also the algorithm increases its time of answer. This answer shows the need to solved these kind of problems on cluster

.

[*] Degree Work. Research Mode.

[**] Faculty of Physics and Mechanics Engineering. School of Computing and Engineering Systems. Director: MSc. Fernando Antonio Rojas Morales. Co-Director: Ph.D. Henry Lamos Díaz.

GLOSARIO

A continuación se presenta una serie de conceptos para el desarrollo de la presente investigación:

- **Función multimodal:** Una función $f(\bar{x})$ es multimodal si dados k intervalos distintos, cumple con la definición de ser unimodal en cada uno de los k intervalos, es decir, posee varios puntos óptimos.

- **Función(es) objetivo:** pueden ser una o varias. Estas funciones se expresan en términos de las variables de decisión, y el resultado de su evaluación es el que se desea optimizar (maximizar o minimizar). Si solo una función es considerada se habla de un problema de optimización mono-objetivo. Si varias funciones son consideradas, el problema se denomina optimización multiobjetivo.

- **Función unimodal:** Una función $f(\bar{x})$ es unimodal en el intervalo $\bar{a} < \bar{x} < \bar{b}$ Si y sólo si es monótona en cada lado del único punto óptimo $\bar{x}^*$ en el mismo intervalo. O sea $\bar{x}^*$ es el único punto mínimo de f en $[\bar{a}, \bar{b}]$.

- **Heurística:** Técnica o procedimiento práctico para resolver problemas.

- **Máquina:** Recurso donde se realiza una operación.

- **Meta-Heurística:** Es una estrategia inteligente para diseñar o mejorar procedimientos heurísticos con un alto rendimiento. Los procedimientos heurísticos utilizan conocimiento acerca de un problema y las técnicas

aplicables, tratando de aportar soluciones o acercarse a ellas haciendo uso de una cantidad de recursos razonable.

- **Modelo determinístico:** Es un modelo matemático donde las mismas entradas producirán invariablemente las mismas salidas, no contemplándose la existencia del azar ni el principio de incertidumbre.

- **Modelo estocástico:** Un modelo es estocástico cuando al menos a una variable del mismo se le incorpora la incertidumbre, es decir, es tomada como un dato al azar. Estos problemas pueden ser resueltos de modo óptimo, aunque con un coste computacional elevado.

- **Objetivos:** Direcciones de mejora de los valores con los que el centro decisor se enfrenta a un determinado problema decisional, pueden ser del tipo maximizar o minimizar. Dentro de los objetivos más comunes para optimizar el JSP se tienen:

  - **Makespan** (C_{max}). Se define como el Max(C_1, C_2,…, C_n). Es equivalente al máximo valor del tiempo de finalización (C_j) del último trabajo en dejar el sistema.

  - **Maximum Lateness** (L_{max}). Se define como el Max(L_1, L_2,…, L_n). Mide el tiempo de tardanza máxima de los trabajos, es decir la peor violación de los tiempos de entrega (d_j).

- **Operación:** Es el procesamiento de un trabajo en una máquina. Cada trabajo está compuesto por una o más operaciones.

- **Restricciones:** expresadas en forma de ecuaciones de igualdad o desigualdad, se deben cumplir o satisfacer para que la solución sea considerada factible, es decir, válida. Si el problema no presenta restricciones, todas las soluciones son válidas.

- **Solución factible:** Solución que satisface todas las restricciones.

- **Solución óptima:** Solución factible que brinda el mejor valor para la función objetivo.

- **Trabajo:** Conjunto de operaciones que se realizan en un conjunto de máquinas. Se identifica como el producto a entregar.

- **Variables de decisión:** contienen los valores que se modifican para resolver el problema.

INTRODUCCIÓN

La optimización puede definirse como aquella ciencia encargada de determinar las mejores soluciones a problemas matemáticos que a menudo modelan una realidad física. El objetivo que se persigue al resolver un problema de este tipo es encontrar una solución óptima con un coste computacional razonable[1].

Los problemas de scheduling implican, en general, asignar una serie de recursos a una serie de tareas, satisfaciendo unas restricciones y uno o varios criterios de optimización. Normalmente dichos recursos son limitados, y por lo tanto las tareas se asignan a los recursos en orden temporal. Desde un punto de vista económico, los recursos limitados se suponen que escasean, y por lo tanto el problema de la planificación de tareas tiene más importancia que la meramente académica[2].

El problema del JSP *(por sus siglas en ingles Job Shop Problem)* consiste en determinar la programación más eficiente para los trabajos que son procesados en varias máquinas[3], cada trabajo tiene un conjunto de operaciones que son procesadas en éstas en un orden especifico. Para n trabajos y m máquinas, el tamaño del espacio solución está dado por la formula $(n!)^m$, por ejemplo, para un problema con 5 recursos y 5 trabajos el número total de planificaciones posibles ya sería de casi 25.000 millones, pero se considera poco práctico el hecho de tomar en cuenta todas las posibilidades para identificar todas las programaciones factibles y finalmente la solución optima[4]. Debido a esta explosión factorial el JSP es considerado un miembro de una gran clase de problemas numéricos complejos

[1] PEREZ, J. 2005. Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo.

[2] Ibid

[3] PINEDO, M, Scheduling: Theory, Algorithms and Systems, Prentice Hall, 1995.

[4] DURAN, R., ROJAS, L., DAZA. V. Un algoritmo genético para el problema de Job Shop Flexible. Revista chilena de ingeniería. Vol. 19 N°1, 2011, pp. 53-61.

de optimización combinatoria conocidos como *NP-Hard (Non Deterministic Polynomial)*[5], donde el espacio solución y el tiempo computacional crecen exponencialmente con el tamaño del problema.

En este aspecto, la optimización numérica ha interesado a los investigadores durante las últimas décadas debido a su presencia en el mundo real, principalmente en los sectores industriales y de servicios, así como su complejidad y a la variedad de situaciones que modela.

Debido a la complejidad de estos problemas, los métodos exactos tienen limitaciones para la solución del JSP en la medida que m y n crezcan, por tanto, se hace necesario implementar métodos heurísticos como el PSO que encuentren de manera eficiente soluciones lo más cercanas posibles al óptimo. Esta es una técnica evolutiva que se usa para solucionar el problema del JSP, de amplia aplicación en diferentes campos, fundamentalmente, para la solución de problemas de optimización continua y también para la solución de problemas combinatorios.

El PSO es un procedimiento que permite optimizar uno o más objetivos simultáneamente, así mismo, debe estar acompañado de la toma de decisiones las cuales conllevan a la obtención de mejores resultados, de tal manera que se haga más eficiente el proceso productivo de las empresas[6].

[5] JAIN, A.S y MEERAN, S. Deterministic Job Shop Scheduling: Past, Present and Future. En: European Journal Of Operational Research, 1999, Vol. 113, pp. 390-434.

[6] PEÑA, V., ZUMELZU, L. Estado del Arte del Job Shop Scheduling Problem. Universidad Técnica Federico Santa María Valparaíso, Chile. Mayo 2006.

1. ESPECIFICACIONES GENERALES

1.1 DEFINICIÓN DEL PROBLEMA Y JUSTIFICACIÓN DEL PROYECTO

En la programación de la producción de las industrias manufactureras se debe decidir sobre la asignación de los recursos a las tareas o trabajos para optimizar, uno o más objetivos en el corto plazo. Para apoyar estas decisiones tradicionalmente se utiliza el modelo de Job Shop. Estos problemas de programación de producción han sido estudiados intensamente en la literatura. Su carácter combinatorio hace difícil su resolución en tiempos razonables mediante métodos exactos debido a su gran complejidad computacional asociada a la categoría NP Hard.

La alternativa a dicho enfoque es utilizar metaheurísticas que intenten resolver el problema mediante la aplicación de criterios de búsqueda, obteniéndose buenos resultados en tiempos computacionales aceptables. Los problemas reales por lo general son multiobjetivos por naturaleza, es decir rara vez una decisión es tomada en base a un solo criterio. Por esta razón, resulta de gran interés el desarrollo de herramientas que reflejen las necesidades industriales y que puedan entregar soluciones en un tiempo razonable, así las metaheurísticas surgen como una alternativa práctica para resolver este tipo de situaciones.

Dicho lo anterior, aplicar el método PSO acompañado de la toma de decisiones, en la resolución de problemas de tipo industrial, y en particular a la resolución de problemas de programación, hace que esta área de la investigación de operaciones tenga una gran importancia en la disminución de costos de producción y en la eficiencia de los sistemas productivos de las empresas. Por lo tanto se hace necesario contar con una buena programación que soporte tomar

decisiones frente a cuál es la mejor combinación de máquinas y trabajos en el momento de producir, permitiendo alcanzar un pleno rendimiento.

1.2 OBJETIVOS

OBJETIVO GENERAL

Desarrollar una herramienta software para la solución del problema JSP (Job Shop Problem) con múltiples objetivos mediante la metaheurística PSO (Particle Swarm Optimization).

OBJETIVOS ESPECÍFICOS

1) Aplicar la variante MOPSO para la solución del problema JSP multiobjetivo.

2) Implementar los modelos de requerimientos UML y el código en JAVA de la aplicación software.

3) Evaluar la herramienta software para un caso particular de 5 máquinas por 5 trabajos y 2 objetivos (makespan y maximum lateness).

2 OPTIMIZACIÓN COMBINATORIA

Existen dos tipos de problemas, los problemas de decisión, en los que se define un conjunto de alternativas o variables de decisión al problema, y los problemas de optimización, que hallan la mejor configuración existente en el conjunto de alternativas.

El primer tipo de problema toma decisiones representadas en variables de decisión, que pertenecen al conjunto de números reales; se utilizan para construir la función objetivo y las restricciones propias del modelo. La función objetivo se denota como $"f(x_1,\ x_2,\ x_3,..,x_n)"$, describe el comportamiento del sistema a través de la relación que existe entre las variables y restricciones que lo definen[7].

Un problema de optimización consiste en encontrar la solución “óptima”, o aquella que encuentre el mejor valor (máximo o mínimo) de la función objetivo. Evidentemente, es más fácil determinar si existe una configuración que cumpla ciertas restricciones, que encontrar la solución óptima, por esta razón, los problemas de decisión son más sencillos de resolver que los problemas de optimización.

2.1 PROBLEMAS COMBINATORIOS

Cuando la función objetivo y las restricciones del modelo son lineales, se presenta un problema de programación lineal, y si las variables de decisión de esos problemas pertenecen a los enteros positivos, el problema de optimización se denomina problema de programación lineal entera. Si a las variables se les incorpora cierta medida de incertidumbre, se tiene un problema de programación

[7] TAHA, Hamdy. Investigación de Operaciones. 7ma edición. Pearson Educación. México, 2004

estocástica. Si las variables son de naturaleza discreta, binaria y continua, se presentan problemas de programación lineal entera mixta.

Cuando las variables se agrupan en varios conjuntos que representan objetos, casos o grafos, la técnica de optimización usada es de carácter combinatorio. La optimización combinatoria se encarga de ubicar dichas variables u objetos en ciertas posiciones. El conjunto de posiciones se llama configuración y la mejor configuración encontrada, es la optimización combinatoria del modelo. En una configuración, las variables están relacionadas de muchas maneras con los demás conjuntos de variables del problema. Un problema de optimización combinatoria se suele representar por medio de un problema de programación lineal.

Según lo anterior, el JSP es un problema de optimización combinatoria, ya que las variables (número de trabajos, número de máquinas) son de naturaleza discreta y se relacionan de muchas maneras unas de otras. La idea es encontrar la mejor configuración de los trabajos a realizar en cada una de las máquinas, de tal manera que se optimice o mejore la función que los define.

2.1.1 COMPLEJIDAD COMPUTACIONAL

El número de interrelaciones de las variables discretas en los problemas combinatorios puede llegar a ser tan grande, que la técnica usada para resolverlo tardaría una cantidad exorbitante de tiempo de cómputo explorando el universo de configuraciones; por eso, el tiempo computacional que usa una técnica para arrojar una respuesta es una condición muy importante a la hora de seleccionar una algoritmo de solución.

Los problemas de orden polinomial o de ejecución en tiempo polinómico, se definen como problemas P, y son problemas tratables en términos de gasto

computacional. De otro lado, los problemas de orden exponencial o NP, son problemas que se resuelven en un tiempo de cómputo no polinómico.

El conjunto de problemas NP son problemas que pueden ser resueltos en tiempo polinómico si se tiene un algoritmo que arroje una solución en tiempo polinómico. Si el algoritmo brinda una solución óptima del modelo en tiempo polinómico, entonces el conjunto de problemas se reduce a orden P, y se tendría un problema NP-Completo. La mayoría de los Problemas de Optimización Combinatoria de interés científico o práctico están incluidos en la clase NP-completos Estos problemas, son los más difíciles de resolver. Un problema es NP-Completo, si cualquier otro problema NP puede ser trasformado en él. Los problemas considerados como los más difíciles en NP-Completo, se definen como NP-Hard.

Los problemas de asignación de recursos son problemas de orden NP-Hard debido a la dificultad que se tiene de enumerar todas las configuraciones de las soluciones, debido a las múltiples interrelaciones de sus variables y el gran tamaño que tienen los datos de entrada del modelo.[8]

2.1.2 TÉCNICAS DE OPTIMIZACIÓN

Debido a la importancia de los problemas de optimización combinatoria en aplicaciones prácticas, científicas e industriales, se han desarrollado múltiples métodos para resolverlos, en los que destacan las técnicas exactas y las aproximadas.

Las técnicas exactas encuentran la solución óptima para cualquier instancia de un problema en un tiempo determinado. Para problemas NP-Hard, el inconveniente

[8] LOZADA, A, CADENA R, Solución del problema de ruteo de vehículos con ventanas de tiempo (VRPTW) mediante métodos heurísticos, Universidad Industrial de Santander, facultad de ingeniería industrial, 2012

de usar estos métodos, es el tiempo de cómputo necesario para determinar una solución óptima. En su lugar, se han desarrollado métodos aproximados que permiten encontrar una solución factible, que en algunos casos puede considerarse cercana a la solución óptima, y que es posible hallarla en un tiempo razonable.

Dentro de los **algoritmos aproximados** se pueden encontrar dos tipos: las heurísticas y las metaheurísticas. En esta clase de problemas, las soluciones que cumplen con las restricciones del modelo, se llaman soluciones factibles del modelo. En el espacio de soluciones factibles, pueden existir uno o varios óptimos locales, que son definidos por el concepto de vecindad o respuestas vecinas de una solución. El óptimo local es aquella solución que es mejor en el vecindario de búsqueda. El conjunto de soluciones factibles puede variar por medio de una operación llamada movimiento, que depende de la naturaleza del problema y de la técnica a usar para la búsqueda de soluciones.
El óptimo global es aquella solución en el que la función objetivo no se puede mejorar a través de los movimientos del conjunto de soluciones en todo el espacio de búsqueda.

Existen problemas para los que no es indispensable hallar la solución óptima; con una buena solución basta para analizar el comportamiento del modelo que se optimiza. Los **métodos heurísticos** permiten hallar esas soluciones, sin embargo hacen que la solución quede atrapada en un óptimo local y no generan movimientos para saltar a otro punto del espacio de búsqueda. Aún así cumplen con las restricciones establecidas para el modelo y mejoran de alguna manera la función objetivo.

En la actualidad, se han desarrollado técnicas de optimización más avanzadas que usan heurísticas de más alto nivel y que permiten la exploración de un mayor

espacio de búsqueda de manera eficiente. Estas herramientas son denominadas metaheurísticas.

Las **metaheurísticas** son métodos aproximados diseñados para resolver problemas de optimización combinatoria, que proporcionan soluciones casi óptimas. Son procedimientos iterativos que guían una heurística dentro de un espacio de soluciones, combinando diferentes conceptos de la inteligencia artificial o la evolución biológica. Estas técnicas son usadas cuando se requiere tomar decisiones en los sistemas complejos o cuando se cuenta con poco tiempo para analizarlos y ponerlos en marcha; en ciertos sistemas logísticos, por ejemplo, es complejo conocer la combinación óptima de los recursos, debido a que se cuenta con poco tiempo o se encuentran múltiples variaciones en los procesos.

3 JOB SHOP PROBLEM (JSP)

El proceso utilizado para determinar la secuencia en la cual se realizan las operaciones de un conjunto de trabajos en las máquinas correspondientes se conoce con el nombre de scheduling[9]. El problema del Job Shop (programación de tareas o talleres), y en general cualquier problema de scheduling, es un problema de optimización combinatoria. La función del scheduling es la asignación de recursos a tareas a lo largo del tiempo y tiene como finalidad la optimización de uno o más objetivos. Los recursos pueden ser máquinas en un taller, pistas en un aeropuerto, ladrillos en una construcción, unidades de procesamiento en un ambiente computacional, etc.[10] Como tareas se pueden tener operaciones de un proceso de producción, despegues y aterrizajes en un aeropuerto, etapas de un proyecto de ingeniería, ejecuciones de un programa computacional, etc. Cada tarea puede tener diferentes niveles de prioridad, así como tiempos de posibles inicios, y los objetivos pueden tomar varias formas, minimizar los tiempos de finalización de la última tarea, minimizar el número de tareas luego de una fecha de entrega acordada, etc.[11]

La programación de tareas apoya a las principales áreas funcionales de una empresa, por ejemplo:

- El departamento de producción necesita conocer la secuencia real de productos a fabricar en el día. Por ello, su personal ha de consultar la programación de tareas que resulta tras la planificación maestra de los diferentes productos a fabricar.

- El departamento de Marketing puede establecer medidas de la eficiencia de la programación para determinar si los tiempos de completación de los

[9] LATORRE, G. “El problema flow shop flexible de dos etapas: programación de las intervenciones quirúrgicas en un hospital”, universidad del bío-bío, facultad de ingeniería. Junio 2011.

[10] SALTO, C. 2000. Algoritmos evolutivos avanzados como soporte del proceso productivo.

[11] PEÑA. Op. cit.

pedidos están aportando una ventaja competitiva y si las entregas se están realizando a tiempo. El conocimiento de los tiempos de entrega de cada pedido permite al departamento de Marketing ofrecer tiempos de entrega más ajustados.

3.1 CLASES DE SCHEDULE

En modelos determinísticos es importante las secuencias o schedules[12]. Hay distintas clases de schedules: nondelay, activo y semiactivo[13].

- Un schedule factible es **nondelay** si ninguna máquina permanece ociosa mientras exista una operación disponible para su procesamiento.

- Un schedule factible es **activo** si no se puede adelantar la finalización de una operación por alteración de la secuencia de procesamiento sin que se retrase otra operación.

- Un schedule factible se llama **semiactivo** si ninguna operación puede finalizar tempranamente sin alterar la secuencia de procesamiento de alguna de las máquinas.

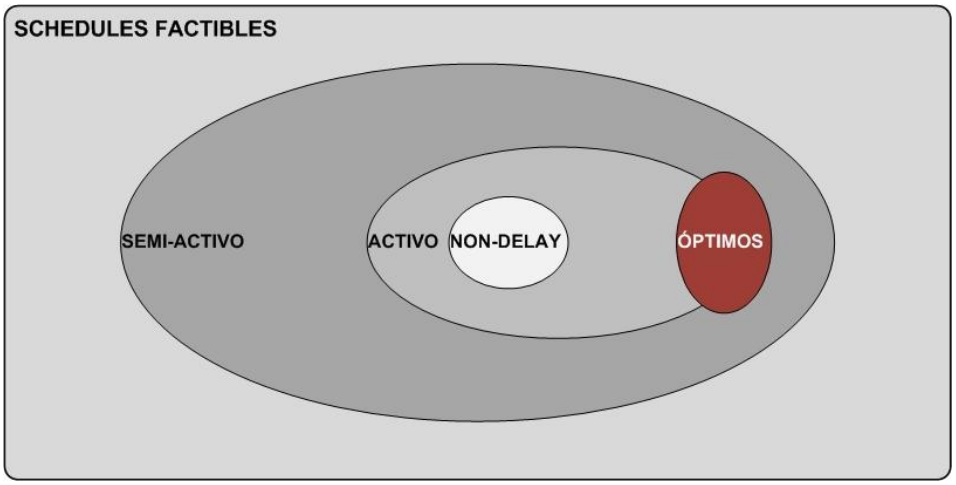
Un schedule óptimo está dentro del conjunto de schedules activos[14]. Un schedule nondelay es activo pero no es cierto lo inverso[15], en la **Figura 1** se puede observar la relación entre las clases de schedules.

---

[12] GIFFLER, B. y. (1960). Algorithms for Solving Production Scheduling Problems. *Operations Research* , vol. 8. nro. 4, pag. 487-503.

[13] FRENCH, S., Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop, Horwood, Chichester, 1982.

Figura 1. Relación entre las clases de schedules



Fuente: Autor

Ejemplo 1. Schedule activo y nondelay

Sea un problema de Job shop con tres máquinas y dos trabajos. El trabajo 1 necesita una unidad de tiempo sobre la máquina 1 y tres sobre la máquina 2. El trabajo 2 necesita dos unidades sobre la máquina 3 y tres sobre la máquina 2 (**Tabla 1**). La máquina 2 es la última en procesar ambos trabajos. El trabajo 2 sobre la máquina 2 se procesa antes que el trabajo 1(**Figura 2**).

Tabla 1. Tiempos de procesamiento para el ejemplo 1

	Operaciones	
	Máquina	**(Tiempo de procesamiento)**
Trabajo	**1**	**2**

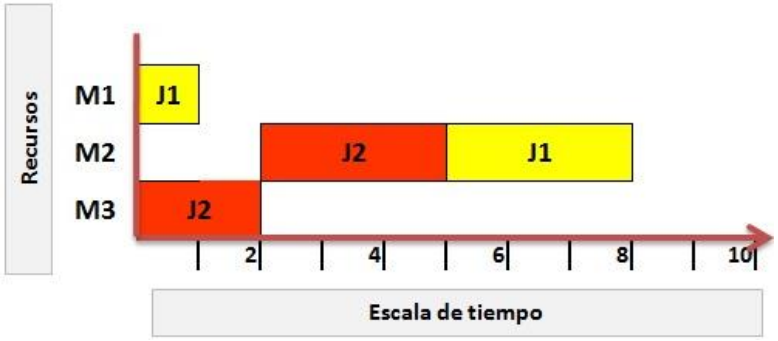
[14] BOOKER, L.B., Improving Search in Genetic Algorithms, en Davis, L. (editor), Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers, pag. 61-73, Los Altos, CA, 1987.

[15] SALTO. Op. cit.

1	1(1)	2(3)
2	3(2)	2(3)

Fuente: Autor

Figura 2. Schedule activo y nondelay



Fuente: Autor

El schedule **no** es un schedule **nondelay**; la máquina 2 permanece ociosa hasta el tiempo 2, mientras que hay un trabajo disponible para su procesamiento en el tiempo 1(**Figura 2**).

Este schedule es **activo**; invirtiendo la secuencia de los dos procesos sobre la máquina 2 se adelanta la finalización del trabajo 1, pero se pospone el procesamiento del trabajo 2 (**Figura 3**).

Figura 3. Schedule activo

Fuente: Autor

Ejemplo 2. Schedule semiactivo

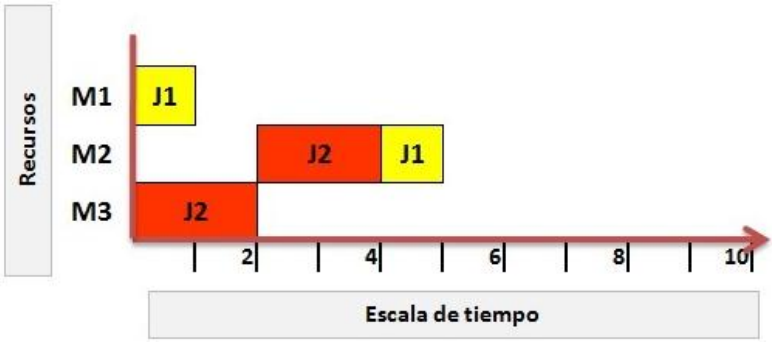
Sea un problema de Job shop con tres máquinas y dos trabajos. Las rutas de los dos trabajos son las mismas que en el ejemplo previo (la máquina 2 es la última en procesar ambos trabajo). El tiempo de procesamiento del trabajo 1 sobre la máquina 1 y la máquina 2 es el mismo e igual a uno. El tiempo de procesamiento del trabajo 2 sobre la máquina 2 y la máquina 3 es de dos (**Tabla 2**). Sea el schedule bajo el cual el trabajo 2 se procesa sobre la máquina 2 antes que el trabajo 1(**Figura 4**). Esto implica que el trabajo 2 comienza su procesamiento sobre la máquina 2 en el tiempo dos y el trabajo 1 comienza su procesamiento sobre la máquina 2 en el tiempo cuatro.

Tabla 2. Tiempos de procesamiento para el ejemplo 2

	Operaciones	
	Máquina (Tiempo de procesamiento)	
Trabajo	**1**	**2**
1	**1(1)**	**2(1)**
2	**3(2)**	**2(3)**

Fuente: Autor

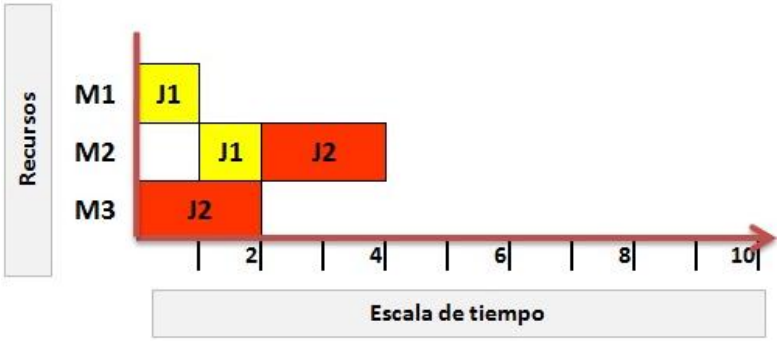
Figura 4. Schedule semi-activo



Fuente: Autor

Este schedule es **semiactivo**; si se adelanta la finalización del trabajo 1, se altera la secuencia de procesamiento sobre la máquina 2. Sin embargo, es **no activo**; el trabajo 1 se puede procesar sobre la máquina 2 sin retrasar el procesamiento del trabajo 2 sobre la misma máquina (**Figura 5**).

Figura 5. Schedule semi-activo y activo



Fuente: Autor

3.2 FORMULACIÓN CLÁSICA DEL JSP

El problema de Job shop scheduling clásico se puede describir como sigue:
Hay un conjunto finito **J** de **n** trabajos ($\mathbf{J_j}$), $\{J_1, J_2, \ldots, J_n\}$, los cuales deben ser procesados en un conjunto finito **M** de **m** máquinas ($\mathbf{M_k}$), $\{M_1, M_2 \ldots, M_m\}$, cada trabajo está compuesto por un conjunto de operaciones $\mathbf{O_{jk}}$ (el procedimiento de un trabajo J_j en una máquina M_k). La operación $\mathbf{O_{jk}}$ requiere el uso exclusivo de

una máquina M_k con una duración ininterrumpida $\mathbf{t_{jk}}$, conocido como el tiempo de procesamiento para una operación **J** en la máquina **k**. Cada trabajo es visto como una secuencia tecnológica de máquinas en las cuales este puede ser procesado.

Existen varias restricciones sobre trabajos y máquinas[16]:

- Un trabajo no puede visitar una misma máquina dos veces.
- No hay restricciones de precedencia entre operaciones de distintos trabajos.
- Las operaciones no se pueden interrumpir.
- Cada máquina puede procesar sólo un trabajo a la vez.
- No se especifican ni release times (fecha de trabajo listo a ser procesado) ni due dates (fecha de entrega).

3.3 MODELO MATEMÁTICO

Muchos investigadores reconocen que los problemas de scheduling pueden ser resueltos óptimamente utilizando técnicas de programación matemática y uno de los modelos matemáticos más comunes del Job Shop Problem es la programación lineal entera mixta, propuesto por Alan Manne en 1960. Se compone de un conjunto de restricciones lineales y una función objetivo lineal, pero con la restricción adicional que algunas variables de decisión $\mathbf{X_{ijk}}$ son enteras. Las variables enteras son binarias y se usan para implementar las restricciones disyuntivas, es decir, que ninguna máquina pueda realizar más de una operación a la vez y ninguna operación pueda ser realizada en más de una máquina al mismo tiempo. M corresponde a un valor arbitrario grande que debe ser mayor que la suma de los tiempos de procesamiento de todas las operaciones menos el tiempo más pequeño.

[16] Ibid.

Función Objetivo:	Minimizar $L=C_{max}$		**(1)**
Sujeto a:			
Restricción de precedencia	$C_{jk}-t_{jk} \geq C_{jh}$	**j**=1,2,…,n; **h,k**=1,2,…,m	**(2)**
Restricción de no traslape	$C_{jk}-C_{ik} + M(1-X_{ijk}) \geq t_{jk}$	**i,j**=1,2,…,n; **k**=1,2,…,m	**(3)**
Condiciones de no negatividad	$C_{ik,}\ C_{jk,}\ t_{jk} \geq 0$	**i,j**=1,2,…,n; **k**=1,2,…,m	**(4)**
Variable binaria	$X_{ijk=}$ 1 ó 0		

Consideraciones:

(2): asegura que la secuencia de procesamiento de las operaciones de cada trabajo corresponda a la orden determinada.

(3): *asegura que cada máquina procese un trabajo a la vez.*

En este modelo existen $2nm + \frac{n(n-1)m}{2}$ variables, de las cuales $\frac{n(n-1)m}{2}$ son binarias. El número de restricciones es $n(m-1) + n(n-1)m + 2nm$. Si n tareas son realizadas por m máquinas, entonces existen potencialmente $(n!)^m$ secuencias, aunque muchas de ellas serán infactibles debido a las varias restricciones. De esta manera, un problema con $n = 20$ y $m = 10$ tiene $7{,}2651\times10^{183}$ posibles soluciones.

3.4 REPRESENTACIÓN DEL JSP

Dentro de las formas más comunes de representación para una solución del JSP se tienen; instancia, diagrama de Gantt y representación mediante grafos. Para mostrar dichas representaciones se introducirá un ejemplo de tamaño 3 x 3.

Ejemplo 3. Representación de un problema JSP

Se tiene un problema de 3 trabajos a ser procesados en 3 máquinas, donde cada trabajo tiene una secuencia correspondiente; el trabajo 1 debe ser procesado en las máquinas 1, 2 y 3. El trabajo 2 en las máquinas 1, 3 y 2, y el trabajo 3 en las máquinas 2, 1 y 3.

- Una **instancia** del JSP se define mediante una matriz, la cual contiene cada uno de los trabajos, el orden dentro de las máquinas y el tiempo de procesamiento correspondiente a cada trabajo en cada máquina[17] (**Tabla 3**)**Tabla 3**.

Tabla 3. Instancia JSP para un problema 3X3

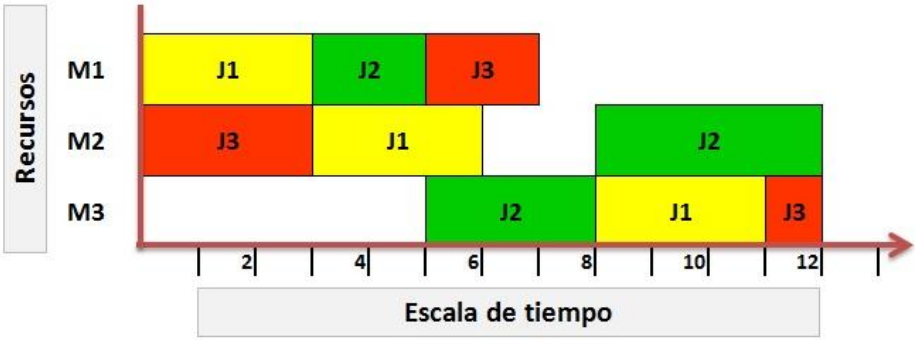
	Operaciones		
	Máquina (Tiempo de procesamiento)		
Trabajo	**1**	**2**	**3**
1	**1(3)**	**2(3)**	**3(3)**
2	**1(2)**	**3(3)**	**2(4)**
3	**2(3)**	**1(2)**	**3(1)**

Fuente: Autor

- El **diagrama de Gantt** es una herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación para diferentes tareas o actividades a lo largo de un tiempo total determinado; representa cada máquina en un renglón diferente y cada cuadro representa una operación, en el eje x se encuentran las unidades de tiempo que gastan los trabajos en completar cada una de las operaciones, por tanto, es una forma conveniente de visualizar una posible solución del JSP (**Figura 6**).

[17] DIAZ, N., LUNA, L. Implementación de un algoritmo inmune artificial aplicado en el área de planificación de recursos. Universidad industrial de Santander. 2012.

Figura 6. Representación de una solución en una gráfica de Gantt para el problema 3X3



Fuente: Autor

- La representación basada en **grafos disyuntivos** propuesta por Tamaki y Nishikawa[18], se considera una clase de representación al problema de Job shop scheduling, se puede representar como G = (N, A, E)[19], donde[20] :

  N: contiene los nodos representando todas las operaciones.

  A: contiene los arcos que conectan operaciones consecutivas del mismo trabajo.

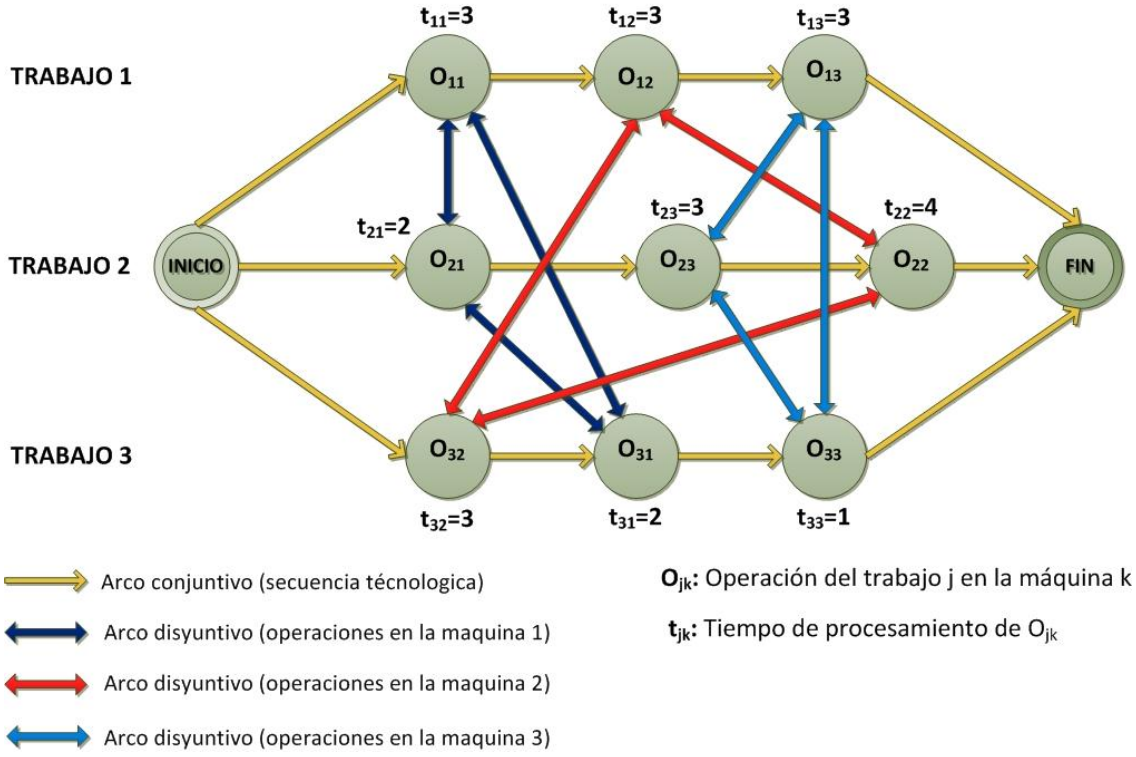
  E: contiene los arcos disyuntos que conectan las operaciones a ser procesadas por la misma máquina.

[18] TAMAKI, H., Mori, M., y Araki, M., Generation of a Set of Pareto-Optimal Solutions by Genetic Algorithms, en Transactions of the Society of Instrument and Control Engineers, vol. 31, nro. 8, pag.1185-1192, 1995.

[19] BALAS, E., Machine Sequencing via Disjunctive Graphs: an Implicit Enumeration Algorithm, en Operations Research, vol. 17, pag. 941-957, 1969

[20] ROY, B., y Sussmann, B., Les Problems d'Ordonnancemente avec Constintes Disjonctives, Technical Report 9, SEMA, Note D.S, Paris, 1964.

Figura 7. Representación de JSP por grafo disyuntivo para el problema de 3X3

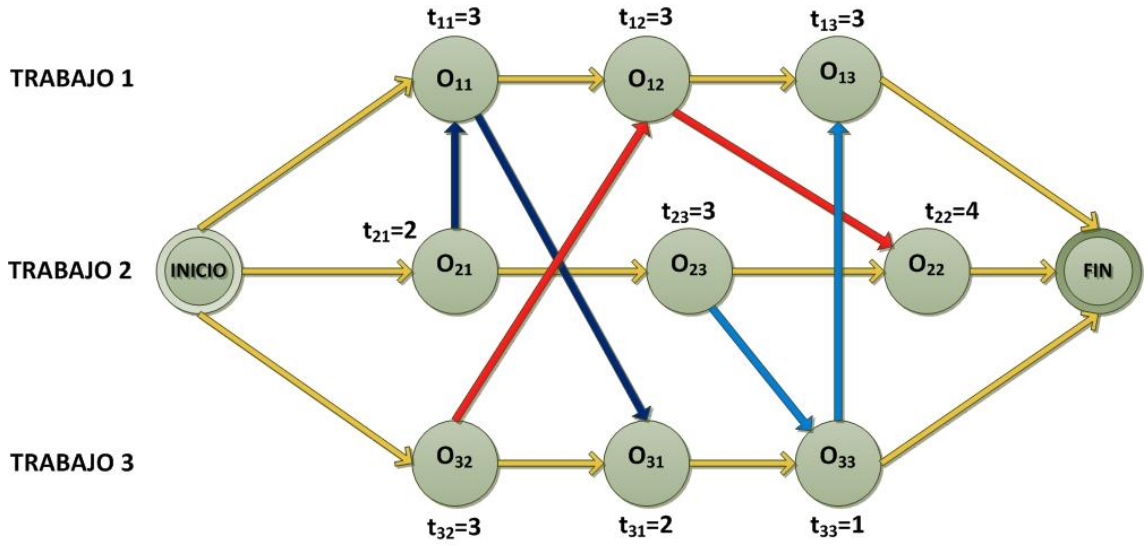


Fuente: Autor

Las restricciones disyuntivas se representan por un eje en E. Se puede establecer un arco disyuntivo por sus dos posibles orientaciones. La construcción de un schedule establece las orientaciones de todos los arcos disyuntivos así como también determina la secuencia de operaciones sobre la misma máquina. La **Figura 7**, muestra el grafo disyuntivo para el ejemplo de tres trabajos y tres máquinas introducido en la **Tabla 3**.

El problema de JSP consiste en encontrar un orden en las operaciones sobre cada máquina, es decir, fijar la orientación de los arcos disyuntivos tal que el grafo resultante no contenga ciclos para garantizar la no existencia de conflictos de precedencia entre las operaciones. Una vez que se determina una secuencia para una máquina se reemplazan los arcos disyuntivos por arcos conjuntivos (**Figura 8**).

Figura 8. Representación por grafo de una solución factible para el problema JSP de 3x3



Fuente: Autor

3.5 JSP CON MÚLTIPLES OBJETIVOS

La mayoría de los estudios de JSP son de un solo objetivo, y dan como resultado un programa para reducir al mínimo el tiempo necesario para completar todos los trabajos, es decir, para minimizar el makespan, sin embargo, la mayor parte de los problemas de optimización del mundo real son naturalmente multiobjetivo y requieren logros simultáneos. Esto significa que la concentración académica de los objetivos del JSP debe ser ampliado de único a múltiples objetivos[21].

El problema de optimización multiobjetivo u optimización vectorial, no puede ser resuelto de la manera clásica debido a que una solución que minimice un objetivo en general no minimizará los demás objetivos. En lugar de hallar una solución

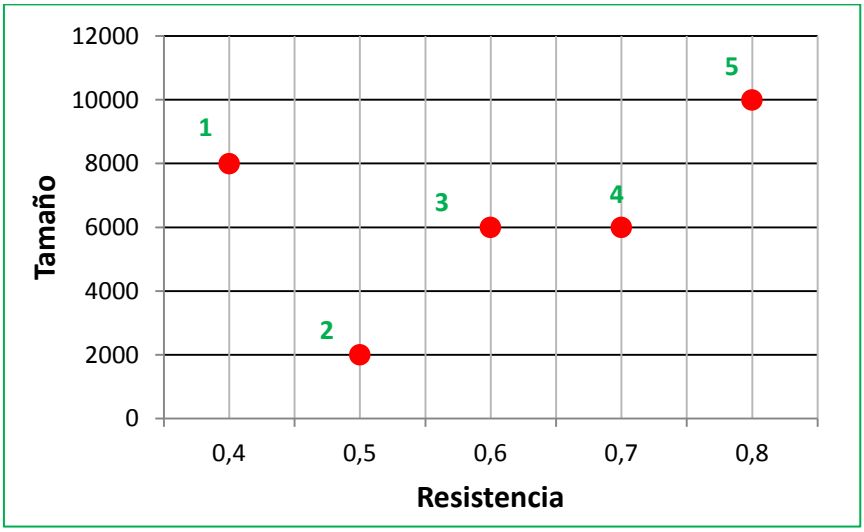
[21] SHA, D. Y., LIN, H. H. A multi-objective PSO for Job Shop-Scheduling Problems. En: Expert Systems with Applications, 2010, Vol. 37, pp. 1065-1070.

óptima se busca una solución o punto eficiente. Una solución **x** es eficiente (o Pareto óptima) cuando es una solución factible (esto es, que cumple las restricciones), tal que no existe otra solución factible **x'** que proporcione una mejora en un objetivo sin producir un empeoramiento en al menos otro de los restantes objetivos. Se puede ver en la **Figura 9** que el punto (opción 4) es un punto eficiente respecto al punto (opción 3), ya que éste mejora el confort, sin causar ningún cambio en el precio (**Ejemplo 4**).

Ejemplo 4. Soluciones óptimas de Pareto.

En una fábrica se pretenden optimizar dos objetivos; el tamaño y la resistencia del producto que fabrican, siendo un tamaño pequeño y una alta resistencia, lo más buscado por los clientes a la hora de escoger su producto. En la **Figura 9** se muestran 5 posibles opciones para optimizar estos dos objetivos.

Figura 9. Soluciones factibles para el ejemplo 4



Fuente: Autor

Una solución factible **x** **domina** una solución factible **x'** para un problema con varios objetivos si **x** es por lo menos tan buena como **x'** con respecto a cada objetivo y es estrictamente mejor que **x'** con respecto a por lo menos un objetivo.

El conjunto de soluciones factibles eficientes no dominadas se llama **conjunto óptimo de Pareto**.

De la **Figura 9** se puede considerar que:

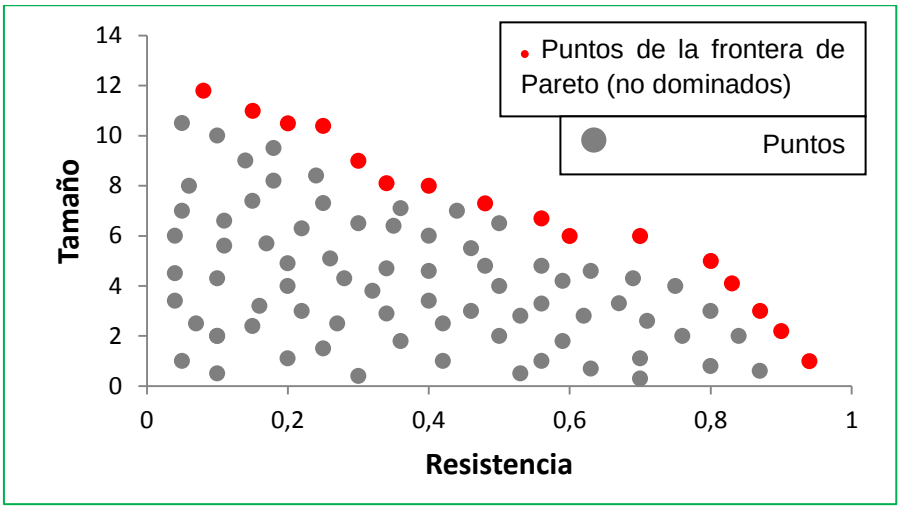
- **La opción 5** es la que ofrece mayor resistencia pero es la de mayor tamaño.
- **La opción 1** es la que ofrece menor resistencia.
- **La opción 2** es la que ofrece menor tamaño. A su vez se considera mejor que la opción 1 puesto que ofrece mayor resistencia y su tamaño es menor.
- **La opción 3** es claramente mejor que 1; ofrece mayor resistencia en un menor tamaño.
- **La opción 4** es mejor que las opciones 1 y 3.

Teniendo en cuenta lo anterior, es posible establecer una relación de dominancia entre las distintas opciones que se muestran:

- 4 domina a 3 y a su vez 3 domina a 1. {4 > 3 > 1 }
- 2 domina a 1. {2 > 1 }

La gráfica del conjunto de soluciones óptimas de Pareto se conoce como frente de Pareto o **frontera de Pareto** (ver **Figura 10**).

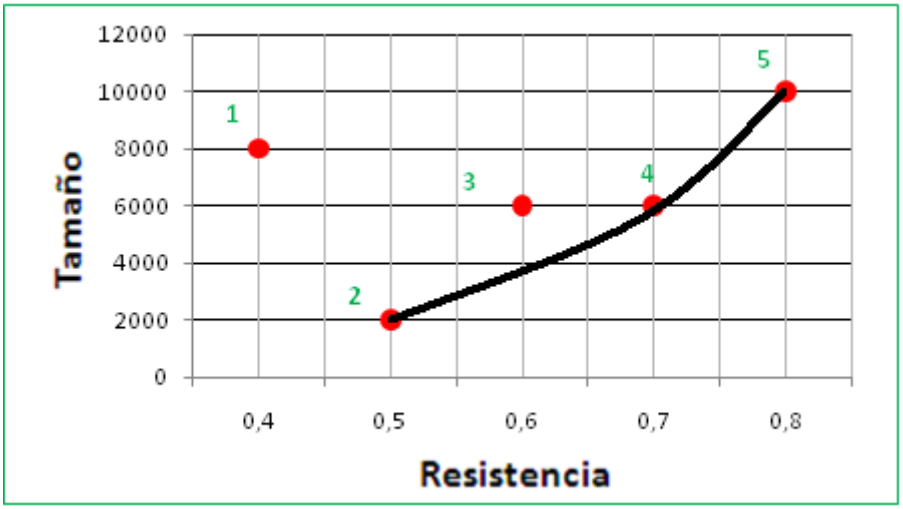
Figura 10. Frontera de Pareto



Fuente: Autor

Siguiendo el ejemplo, la frontera de Pareto está conformada por las opciones {2, 4 y 5} (**Figura 11**).

Figura 11. Frontera de Pareto para el ejemplo 4



Fuente: Autor

Puede decirse que la eficiencia Paretiana es una condición exigida como necesaria para poder garantizar la racionalidad de las soluciones generadas por los diferentes enfoques multiobjetivo, en los cuales no suele existir una única

solución óptima, sino un conjunto (a veces infinito) de soluciones no dominadas que forman la frontera de Pareto.

La operatividad de la programación multiobjetivo consiste entonces en desarrollar como primer paso, una serie de técnicas que permitan a partir de la siguiente estructura **(5)**, generar el conjunto de soluciones factibles y eficientes o Pareto óptimas.

$$\text{EFF}\, f(x) = \left(f_1(x), f_2(x), \ldots f_n(x)\right) \qquad \textbf{(5)}$$

$$s.a \quad x \in X$$

Donde:

- $\mathbf{EFF}$ significa la búsqueda de soluciones eficientes o Pareto óptimas.
- $\boldsymbol{f_i(x)}$ es la expresión matemática de cada objetivo.
- $\boldsymbol{x}$ es el vector de variables de decisión.
- $\boldsymbol{X}$ es el conjunto de restricciones que definen el conjunto de posibles soluciones.

Los métodos más utilizados para generar, o al menos aproximar, el conjunto de soluciones eficientes son: el método de las ponderaciones, restricciones, hibrido entre ponderaciones y restricciones, y el simplex multicriterio.

- **Método de las ponderaciones:** En este método cada objetivo se multiplica por un peso o factor positivo ($\boldsymbol{\lambda}$) que refleja las preferencias de quien toma las decisiones respecto a la importancia relativa de cada meta, procediéndose seguidamente a agregar todos los objetivos ponderados en

una única función objetivo. La optimización de dicha función ponderada y para cada vector de pesos λ se obtiene un elemento del conjunto eficiente. Por ejemplo en un problema con ***n*** objetivos a maximizar, la aplicación del método de las ponderaciones conduce a la siguiente formulación matemática:

$$Max \ \sum_{i=1}^{n} \lambda_i \, f_i(x) \qquad \textbf{(6)}$$

$$s.a \qquad x \ \in X$$

$$\lambda \ \geq 0$$

- **Método de las restricciones o jerarquías:** Este método se basa en la minimización (maximización) de una de las *n* funciones objetivo, que se asume la principal o preferida, y considera a los demás objetivos como restricciones que están acotadas por ciertos niveles permisibles **ε** que corresponden a cotas inferiores. Para cada conjunto de valores que se dé al vector de términos independientes, o término de la derecha (**ε**), se genera un elemento del conjunto eficiente. Así, en un problema multiobjetivo con n objetivos a maximizar, la aplicación del método de las restricciones o jerarquías, conduce al siguiente programa lineal paramétrico:

$$Max \ f_j(x) \qquad \textbf{(7)}$$

$$s.a \qquad x \ \in X$$

$$f_j(x) \ \geq \ \varepsilon_i \quad i = 1, 2, \dots j-1, j+1 \dots n$$

Variando los términos de la derecha $\boldsymbol{\varepsilon_i}$ se van generando los elementos del conjunto eficiente.

- **Método hibrido entre ponderaciones y restricciones:** Este método como su nombre lo indica es una combinación del método de restricciones en el

cual del conjunto de funciones a optimizar se escoge una única función objetivo, y las restantes son incorporadas al conjunto de restricciones del problema, donde cada una de estas es multiplicada por un factor de ponderación λ.

- **Método simplex multicriterio:** El Simplex multicriterio genera todos los puntos de esquina (corner points) eficientes de un problema multiobjetivo desplazándose para ello de un punto esquina al punto esquina contiguo. El algoritmo del Simplex tradicional constituye el mecanismo adecuado para efectuar este tipo de desplazamiento de un punto de esquina a un punto adyacente, por medio de una operación de pivotado. En combinación con esta operación de salto de un punto de esquina a otro, el Simplex multicriterio recurre a una subrutina que permite comprobar la eficiencia o no de cada punto obtenido. El Simplex multicriterio trabaja eficientemente sólo con problemas de un tamaño reducido. Entendiendo por tamaño reducido, problemas con un número de objetivos inferior a cinco, así como un número de variables y restricciones no superior a cien.

Con la aplicación de alguno de los métodos anteriormente mencionados se consigue dividir el conjunto factible en dos subconjuntos: el subconjunto de soluciones eficientes y el subconjunto de soluciones dominadas o inferiores.[22]

[22] CARLOS ROMERO, Análisis De Las Decisiones Multicriterio. Ingeniería de Sistemas. Isdefe, 1996.

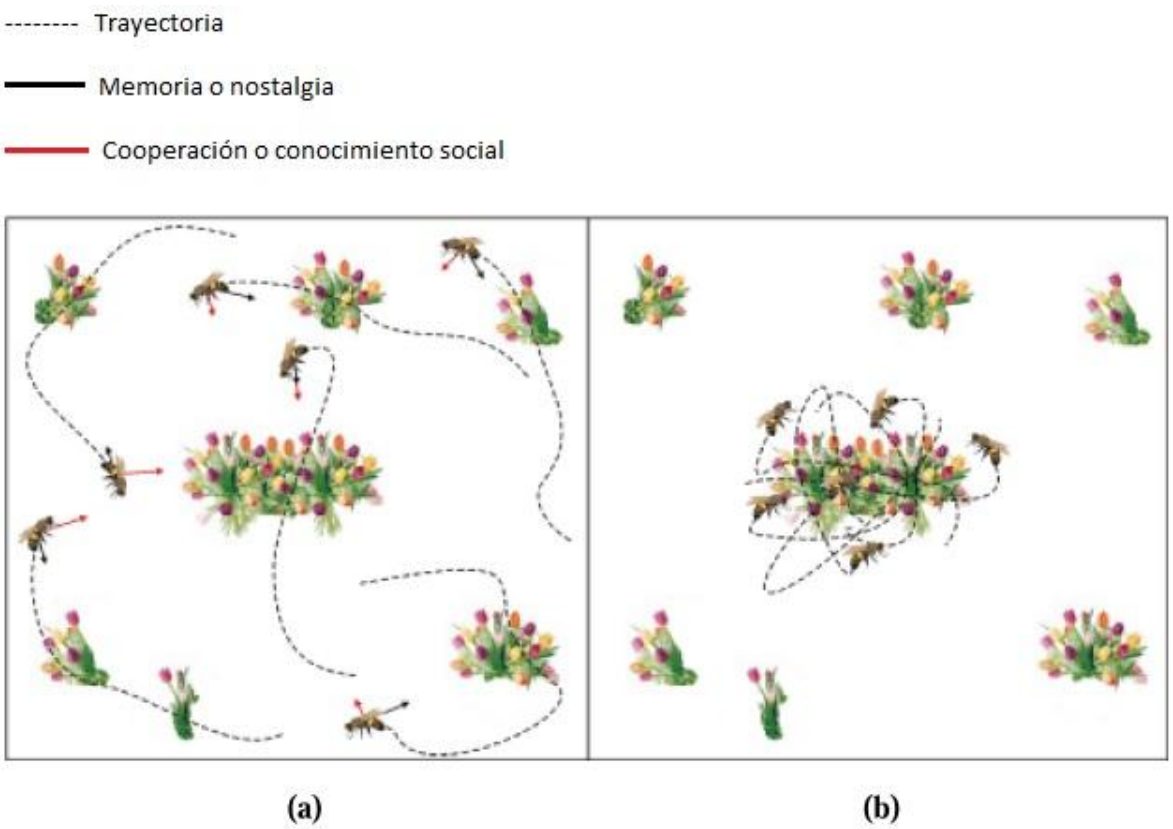
4 PARTICLE SWARM OPTIMIZATIÓN

La optimización con enjambre de partículas, más conocida en la literatura científica como Particle Swarm Optimization (PSO), nace en un intento por imitar y mimetizar el comportamiento social de las bandadas de pájaros o bancos de peces, a partir de la interacción de los individuos entre sí y con el entorno. El PSO fue propuesto por primera vez por Kennedy y Eberhart en 1995[23], parte de la hipótesis de que cada partícula o agente que representa a los peces, pájaros, abejas, hormigas o cualquier otro tipo de individuos que exhiban un comportamiento social como grupo, se lanza a volar en el espacio de búsqueda guiada por la partícula que mejor solución ha encontrado hasta el momento y que cumple con la función de líder de la bandada. Las partículas evolucionan teniendo en cuenta la mejor solución encontrada en su recorrido y la del líder.

De acuerdo con los fundamentos teóricos del método, el movimiento de cada una de estas partículas hacia un objetivo común está condicionado por dos factores básicos, la memoria autobiográfica de la partícula o nostalgia y la influencia social de todo el enjambre. La posición instantánea de cada una de las partículas de la población en el espacio N-dimensional representa dentro del dominio de la función objetivo que se propone, una posible solución, siendo N el número de incógnitas del problema. Básicamente, el proceso evolutivo se reduce a mover cada partícula dentro del espacio de soluciones con una velocidad que variará de acuerdo a su velocidad actual, a la memoria de la partícula y a la información global que comparte el resto del enjambre, utilizando una función de fitness***** para cuantificar la calidad de cada partícula en función de la posición que ésta ocupe.

Figura 12. Modelado del PSO utilizando como símil el movimiento de un enjambre de abejas

[23]KENNEDY, J., EBERHART, R. Particle Swarm Optimization.Proc. IEEE Int. Conf. Neural Networks, 39-43, 1995.



Fuente: PEREZ, JESÚS, 2005.

De la **Figura 12:**

(a) En su desplazamiento, las abejas son atraídas hacia las zonas de mayor concentración de flores encontradas personalmente por cada individuo (memoria) y por el conjunto del enjambre (cooperación).

(b) Una vez que las abejas han sido atraídas a la zona con mayor concentración de flores, que equivale en términos de PSO a la convergencia hacia una solución global, éstas permanecen sobrevolando dicha zona con velocidades muy reducidas.

4.1 ESTRUCTURA DE LA PARTICULA

En una población de **I** partículas, cada partícula del enjambre consta de los siguientes componentes:

- Un vector $\overline{\boldsymbol{x}}_{\boldsymbol{i}} = [\overline{\boldsymbol{x}}_{\boldsymbol{i1}}, \overline{\boldsymbol{x}}_{\boldsymbol{i2}}, \ldots\ldots, \overline{\boldsymbol{x}}_{\boldsymbol{in}}]$ que describe la ***posición de la partícula*** dentro del espacio de soluciones. El tamaño de este vector depende del número de variables necesarias para resolver el problema. Cada posición de este vector se denomina dimensión. $\overline{\boldsymbol{x}}_{\boldsymbol{i}}$ corresponde a una solución potencial al problema de optimización.

- Un vector $\overline{\boldsymbol{v}} = [\overline{\boldsymbol{v}}_{\boldsymbol{i1}}, \overline{\boldsymbol{v}}_{\boldsymbol{i2}}, \ldots\ldots, \overline{\boldsymbol{v}}_{\boldsymbol{in}}]$ que representa la ***velocidad de la partícula***. Este vector al igual que $\overline{\boldsymbol{x}}$ se modifica en el tiempo, reflejando así el cambio de dirección que sufre la partícula dentro del espacio de búsqueda.

- ***pbest o influencia cognitiva*** $\boldsymbol{p}_{\boldsymbol{i}} = [\boldsymbol{p}_{\boldsymbol{i1}}, \boldsymbol{p}_{\boldsymbol{i2}}, \ldots\ldots, \boldsymbol{p}_{\boldsymbol{in}}]$, mejor valor de posición encontrado por la partícula hasta el momento, es decir, cada partícula mantiene en memoria información de la posición espacial asociada con la mejor solución históricamente visitada por ésta.

- ***gbest o influencia social*** $\boldsymbol{G}_{\boldsymbol{i}} = [\boldsymbol{g}_{\boldsymbol{1}}, \boldsymbol{g}_{\boldsymbol{2}}, \ldots\ldots, \boldsymbol{g}_{\boldsymbol{n}}]$ mejor valor de posición encontrado por alguna partícula del enjambre, es decir, cada partícula también conoce la mejor posición o solución encontrada por todos sus congéneres.

- Un valor objetivo o ***aptitud fitness***, representa la calidad de la solución dada por el vector $\overline{\boldsymbol{x}}$, obtenido a través del cálculo de la función de evaluación correspondiente al problema específico.

4.2 TRAYECTORIA DE LA PARTICULA

La trayectoria de la partícula es la que regula la exploración del espacio de búsqueda, está definida con base en el vector de velocidades $\overline{\boldsymbol{v}}$ y el de posición $\overline{\boldsymbol{x}}$, los cuales son sumados para obtener la nueva dirección. Cada partícula recorre el espacio ayudada por dos valores adicionales: el pbest y el gbest. Los cambios de trayectoria son los que determinan la característica principal del algoritmo PSO, ya que a través de los mismos las partículas son "guiadas" a buscar soluciones en las áreas más promisorias del espacio de búsqueda. Si los valores de $\overline{\boldsymbol{v}}$ no se modificaran, la partícula sólo se movería con pasos uniformes en una única dirección.

Cabe destacar que el movimiento del enjambre se realiza en pasos temporales, que se traducen a nivel de algoritmo en iteraciones contiguas. En cada iteración ***k*** del método, cada una de las partículas de la población recorre el espacio de soluciones con una velocidad $\boldsymbol{v_i}$ hacia nuevas posiciones $\boldsymbol{x_i}$, de acuerdo con su propia experiencia $\boldsymbol{p_i}$, y con la experiencia aportada por el mejor de sus convecinos $\boldsymbol{G}$. Esta formulación se reduce a las siguientes ecuaciones[24]:

$$v_{in}(k+1) = v_{in}(k) + C_1 r_1(k) * \left(p_{in}(k) - x_{in}(k)\right) + C_2 r_2(k) * \left(g_n(k) - x_{in}(k)\right) \quad \textbf{(8)}$$

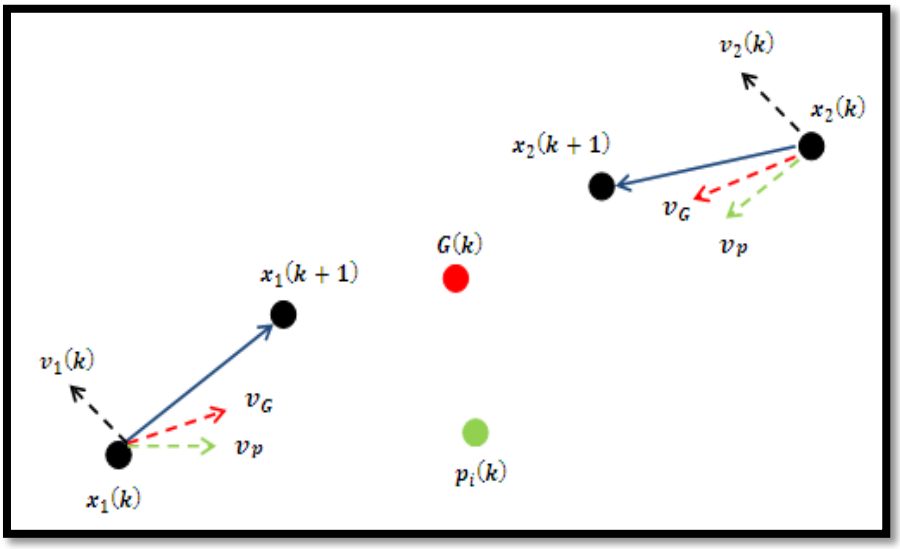
$$x_{in}(k+1) = x_{in}(k) + v_{in}(k+1) * \Delta t \quad \textbf{(9)}$$

[24] ZHENG, Y. M. (2003). On the convergence analysis and parameter selection in particle swarm optimization. *Proceedings of the International Conference on Machine Learning and Cybernetics* , Vol. 3, pp. 1802-1807.

El movimiento de los agentes sobre el espacio de soluciones y, en consecuencia, el rendimiento del algoritmo, está condicionado por el grado de contribución de las tres componentes de la velocidad en **(8)** que reflejan el intercambio de información y el comportamiento social como grupo de las partículas de la población.

La **Figura 13** ilustra la obtención de la nueva posición $x_{in}(k+1)$ de la partícula $x_{in}(k)$ como consecuencia de la suma de la velocidad y las memorias de los mejores.

Figura 13. Trayectoria de una partícula



Fuente: Autor

4.3 PARÁMETROS DE PSO

Los factores $\boldsymbol{C_1}$ y $\boldsymbol{C_2}$ de **(8)** son las denominadas constantes de aceleración cognitiva y social[25], que determinan en qué medida influyen sobre el movimiento de la partícula su propia memoria y la cooperación entre individuos, respectivamente.

Los términos $\boldsymbol{r_1(k)}$ y $\boldsymbol{r_2(k)}$ son dos números aleatorios uniformemente distribuidos en el rango [0,1], cuyo objetivo es emular el comportamiento estocástico y un tanto impredecible que exhibe la población del enjambre[26].

Como puede observarse en la fórmula **(8)**, un incremento de $\boldsymbol{C_2}$ sobre $\boldsymbol{C_1}$ ($\boldsymbol{C_2} > \boldsymbol{C_1}$) aumenta la influencia del valor gbest (la partícula confía más en los resultados de la búsqueda del enjambre que en los propios), lo cual resulta en una mayor *exploración* del espacio (la *exploración* considera las diferentes regiones del espacio de búsqueda a fin de encontrar buenas soluciones)[27], en cambio, cuando $\boldsymbol{C_1} > \boldsymbol{C_2}$ causa que la partícula se mueva en dirección más cercana a pbest (la partícula tendrá más confianza en sus resultados de búsqueda que en los del resto del enjambre), lo cual resulta en una mayor *explotación* del espacio[28] (la *explotación* es la habilidad que el algoritmo posee de concentrarse en una porción del espacio que sea promisoria con el fin de encontrar posiblemente el óptimo[29].

---

[25] Ibid
[26] Ibid
[27] CORREA, R., BEGAMBRE, O., CARRILLO, J. Validación de un algoritmo híbrido del PSO con el método simplex y de topología de evolución paramétrica. UIS, 2010.

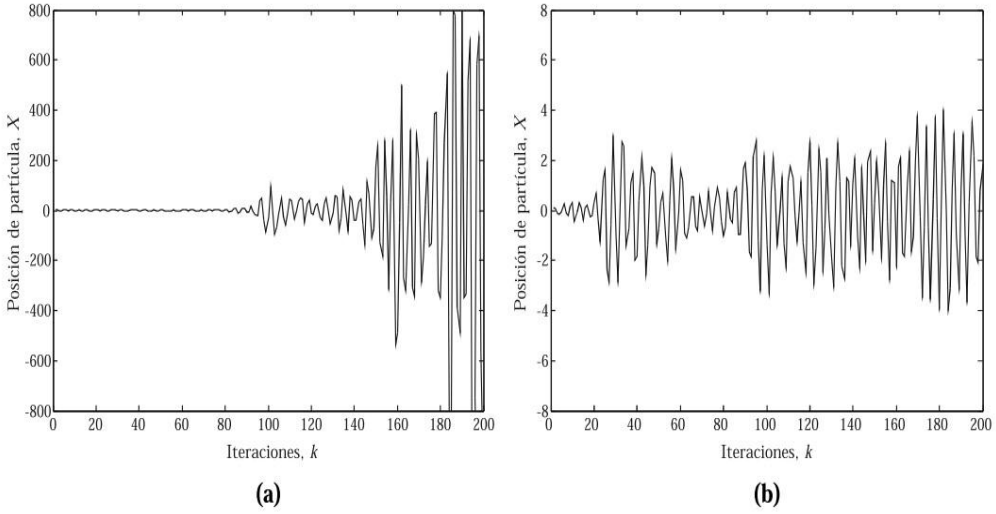
[28] XU, S., RAHMAT, Y. Boundary conditions in particle swarm optimization revisited, IEEE Trans. Antennas Propagat., Vol. 55, no.3, 760-765, Mar., 2007.

[29] CAGNINA, L. Optimización Mono y Multiobjetivo a través de una Heurística de Inteligencia Colectiva. Doctorado en Ciencias de la Computación, Universidad Nacional de San Luis. Argentina, 2010.

Después de calcular la nueva velocidad de la partícula $\boldsymbol{i}$ en la dimensión n, la nueva posición $x_{in}(k+1)$ se actualiza de acuerdo con **(9)**, donde se asume que la velocidad se aplica durante un cierto período de tiempo $\Delta\boldsymbol{t}$, típicamente de valor unitario. El proceso descrito se extiende al espacio N-dimensional, de forma que se van actualizando iterativamente nuevos vectores de posición $\boldsymbol{x_i}$, $\boldsymbol{p_i}$ y $\boldsymbol{G}$, utilizando una función de fitness para ponderar la calidad de dicha solución parcial.

Uno de los objetivos fundamentales de toda heurística es lograr un balance entre la explotación y exploración del espacio de búsqueda. En PSO esta tarea recae en la velocidad de las partículas. Para acotar esta velocidad e impedir que las partículas escapen del espacio de búsqueda del problema, se especifica un valor máximo v_{max} que restringe la velocidad en cada dimensión al intervalo $[-v_{max}, v_{max}]$. Si el valor de v_{max} es demasiado grande las partículas pueden sobrepasar e ignorar continuamente la zona con la solución global. En el extremo opuesto, si v_{max} toma valores extremadamente pequeños las partículas explorarán el espacio de soluciones muy lentamente y podrán quedar atrapadas alrededor de soluciones locales. Con el fin de demostrar el efecto de v_{max}, en la **Figura 14(a)** se representa el desplazamiento en una dimensión de una partícula aislada sin control de velocidad, en su intento por converger hacia el punto $p = g = 0.0$. Si bien el límite de la velocidad v_{max} **Figura 14(b)** evita la divergencia de la partícula, su efecto se limita a contener el paso de la velocidad, pero en realidad no promueve la convergencia hacia el punto óptimo.

Figura 14. Explosión del modelo original de PSO en una dimensión



Fuente: PEREZ, JESÚS, 2005.

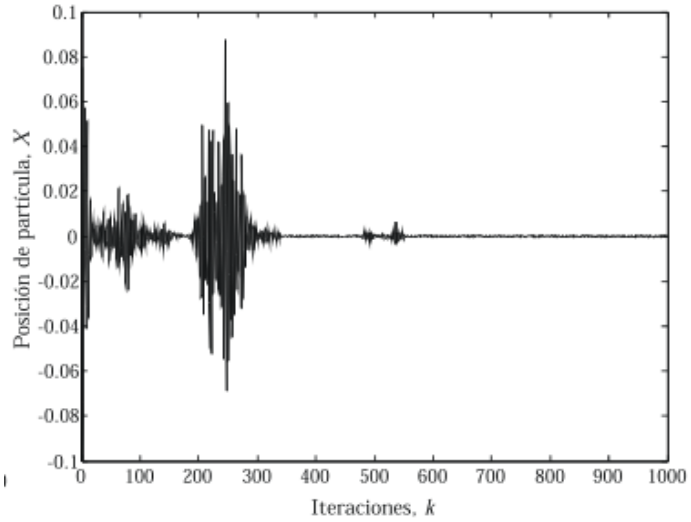
Con el objetivo de reducir el efecto de v_{max} y perfeccionar el control del alcance de la búsqueda sobre el espacio de soluciones, surgen versiones mejoradas del algoritmo que incorporan el concepto de peso inercial ($\boldsymbol{W}$). El factor de peso inercial, variable en el tiempo, controla la influencia de la velocidad previa de la partícula sobre la velocidad actual. Un valor alto de $\boldsymbol{W}$ facilita la exploración global del espacio de búsqueda evitando que las partículas queden atrapadas en óptimos locales, mientras que un valor pequeño posibilita realizar una búsqueda local. De esta forma con la actualización de la matriz de velocidad, las partículas evolucionan de acuerdo con (10):

$$v_{in}(k+1) = W * v_{in}(k) + C_1 r_1(k) * \left(p_{in}(k) - x_{in}(k)\right) + C_2 r_2(k) * \left(g_n(k) - x_{in}(k)\right) \quad \textbf{(10)}$$

En la **Figura 15** se muestra la trayectoria que describe una partícula de acuerdo con el modelo inercial, utilizando $\boldsymbol{W} = \mathbf{0.8}$ y sin límite v_{max}. Se resalta la capacidad de convergencia que introduce el peso inercial. Con el transcurso de las iteraciones, el enjambre limita su movimiento a áreas pequeñas en las

inmediaciones de la solución global G, no obstante, la aleatoriedad del algoritmo permite a las partículas explorar otras zonas del espacio en busca de nuevas soluciones, pero el peso inercial rápidamente redirige la búsqueda hacia el punto óptimo.

Figura 15. PSO con peso inercial



Fuente: PEREZ, JESÚS, 2005.

Es posible reducir el número de iteraciones del algoritmo PSO al encontrar un equilibrio entre las capacidades de búsqueda local y global[30]. Con el fin de mantener este equilibrio usualmente se utiliza un rango de valores para $\boldsymbol{W}$ entre [0,1] para regular la relación entre exploración y convergencia[31], por lo general este factor decrece linealmente desde $W_{max} = 0.9$ al inicio del proceso de optimización, hasta $W_{min} = 0.4$ al final del proceso[32].

[30] PEREZ. Op. cit.

[31] CORREA. Op. cit.

[32] XU. Op. cit.

Luego, mediante la fórmula **(10)**, el factor de inercia es actualizado en cada paso o iteración (k) hasta el final de las iteraciones (K_T)[33].

$$W = W_{max} - \frac{W_{max} - W_{min}}{K_T} * k \qquad \textbf{(11)}$$

Si el número máximo de iteraciones (K_T) es demasiado grande, el PSO puede quedarse estancado a la espera de que el peso inercial decrezca para converger hacia las regiones de interés. Por el contrario, un número excesivamente reducido de iteraciones puede forzar la convergencia prematura hacia una solución local. Paralelamente al peso inercial decreciente, en[34] se propone una nueva alternativa basada en utilizar un peso inercial creciente **(12)** para eliminar en cierta medida la dependencia del número de iteraciones y combinar exploración y convergencia;

$$W = 0.5 + \frac{rnd}{2} \qquad \textbf{(12)}$$

Donde: $\boldsymbol{rnd}$ representa un número aleatorio con distribución uniforme [0,1].

[33] CENTENO, A., AGUILERA, A. Problemas de Optimización en Ingeniería y el Algoritmo de Enjambre de Partículas Departamento de Física, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela. Revista Ingeniería UC, Vol. 16, Nº. 1, Abril 2009 59 - 64

[34] Y-L. ZHENG, L-H. Ma, L-Y. Zhang, J-X. Qian, "On the convergence analysis and parameter selection in particle swarm optimization", Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Xi'an (China), November 2003, Vol. 3, pp. 1802-1807.

5 PSO APLICADO AL JSP

En esta investigación, se utiliza la técnica evolutiva de optimización con enjambre de partículas (MOPSO) propuesta en[35], para resolver el JSP con múltiples objetivos.

El Job Shop Problem se puede convertir en un problema de optimización continua mediante la construcción de la relación correspondiente entre un vector real y un cromosoma obtenido usando la regla de representación basada en el método de prioridad. El mejor esquema de solución se garantiza que sea un schedule activo. Los resultados de Zhang, Li, Li y Hang (2005) demostraron que la permutación basada en la representación de la posición supera a la representación basada en prioridades. Hecha la elección de realizar una representación de posición basada en permutación, también hay que ajustar la velocidad de la partícula y el movimiento de esta. Además, se propone también el mantenimiento de óptimos de Pareto y un procedimiento de diversificación para lograr un mejor rendimiento.

El PSO original fue diseñado para una solución de espacio continuo, por lo que se hace necesario modificar la representación de la posición, velocidad, y movimiento de partículas, para que funcionen mejor con los problemas de scheduling cuyo espacio de solución es discreto. A continuación de describen dichos cambios.

5.1 POSICIÓN DE LA PARTÍCULA

Para representar la posición de la partícula, se genera al azar un grupo de partículas (soluciones) representado por una secuencia de permutación que es una lista ordenada de operaciones (lista de preferencia). Para un problema de n

[35] SHA. Op. cit.

trabajos, m máquinas, la posición de la partícula k puede ser representada por una matriz m x n, como sigue:

$$x^k = \begin{bmatrix} x_{11}^k & x_{21}^k & \cdots & x_{n1}^k \\ x_{12}^k & x_{22}^k & \cdots & x_{n2}^k \\ \vdots & \vdots & & \vdots \\ x_{1m}^k & x_{2m}^k & \cdots & x_{nm}^k \end{bmatrix}$$

Donde $\boldsymbol{x_{ij}^k}$ denota la prioridad de la operación $\boldsymbol{o_{ij}}$, lo que significa que la operación de trabajo $\boldsymbol{i}$ debe ser procesada en la máquina $\boldsymbol{j}$. Es posible decodificar una posición de la partícula en un programa activo empleando la heurística de Giffler y Thompson (1960).

El algoritmo de Giffler y Thompson **(G&T)**[36] se describe brevemente a continuación.

Notación:

- $(\boldsymbol{i,j})$ es la operación del trabajo i que debe ser procesada en la máquina j.
- $\boldsymbol{S}$ es el schedule parcial que contiene las operaciones programadas.
- $\boldsymbol{\Omega}$ es el conjunto de operaciones que se pueden programar.
- $\boldsymbol{s}_{(i,j)}$ es el momento en el que la operación $(\boldsymbol{i,j})$ perteneciente a $\boldsymbol{\Omega}$ se puede iniciar.
- $\boldsymbol{p}_{(i,j)}$ es el tiempo de procesamiento de la operación $(\boldsymbol{i,j})$.
- $\boldsymbol{f}_{(i,j)}$ es el momento en el que la operación $(\boldsymbol{i,j})$ perteneciente a $\boldsymbol{\Omega}$ puede ser terminada, $\boldsymbol{f}_{(i,j)} = \boldsymbol{s}_{(i,j)} + \boldsymbol{p}_{(i,j)}$

[36] SHA Ibid

Algoritmo G&T:

Paso 1: Inicializar $S = \emptyset$ y Ω que contiene todas las operaciones sin predecesores.

Paso 2: Determinar $f^* = min_{(i,j)\in\Omega}\{f_{(i,j)}\}$, es decir el mínimo tiempo de finalización de las operaciones que pertenecen a Ω, y designar como m^* a la máquina en la que se realiza f^*.

Paso 3:

1) Identificar el conjunto de operaciones $(i', j') \in \Omega$ tal que (i', j') requieren también la máquina m^*, y cumplan con $s_{(i,j)} < f^*$

2) Elegir la operación (i, j) del conjunto identificado en el paso **3(1)** con la mayor prioridad, teniendo en cuenta que 1>2>…> n

3) Añadir la operación (i, j) a S.

4) Asignar $s_{(i,j)}$ como el tiempo de finalización $f_{(i,j)}$ de (i, j).

Paso 4: Si se ha generado un schedule completo, detener. De lo contrario, borrar (i, j) de Ω, e incluir su sucesor inmediato en Ω y, a continuación, volver al paso **2**.

Para ilustrar el algoritmo **G&T** se plantea la siguiente instancia para un problema 3x3.

Tabla 4. Ejemplo de un problema de JSP 3x3

trabajo	Secuencia de máquinas	Tiempos de procesamiento
1	1→2→3	P_{11}=3 P_{12}=3 P_{13}=3
2	1→3→2	P_{21}=2 P_{22}=4 P_{23}=3
3	2→1→3	P_{31}=3 P_{32}=3 P_{33}=1

Fuente: Autor

La posición de la partícula $k = 1$ se representa por una matriz $m \times n$:

$$x^1 = \begin{matrix} J_1 & J_2 & J_3 \end{matrix} \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix} \begin{matrix} m_1 \\ m_2 \\ m_3 \end{matrix}$$

Inicialización del algoritmo G&T:

Paso 1.

- $S = \emptyset$ (empieza vacio)
- $\Omega = \{(1,1), (2,1), (3,2)\}$

.

.

Paso 4: Como no se ha generado un schedule completo, se borra la operación escogida, de Ω, y se incluye su sucesor inmediato de acuerdo con la matriz de operaciones a realizar, el nuevo conjunto Ω:

$$\Omega = \{(1,2), (2,1), (3,2)\}$$

… Iteración 9…

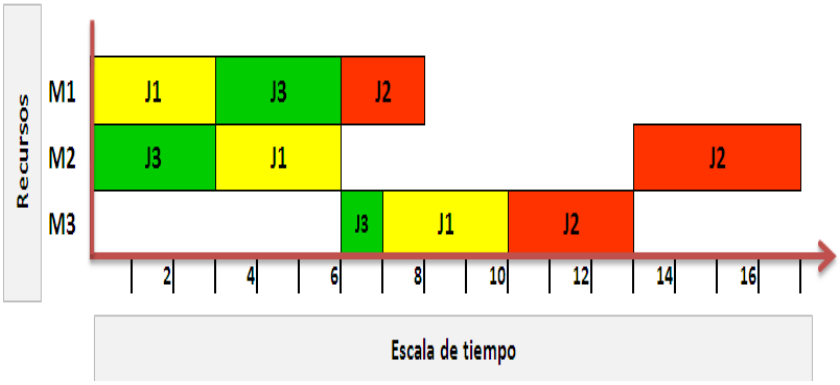
1) $S = \{(1,1), (3,2), (3,1), (1,2), (3,3), (2,1), (1,3), (2,3), (2,2)\}$ ver **Figura 16**.

Paso 4: Se ha generado un schedule completo, entonces el algoritmo se detiene.

$$\boldsymbol{\Omega} = \{\emptyset\}$$

Al final de las iteraciones es creado el schedule completo, el algoritmo se muestra en detalle en el **Anexo A.**

Figura 16. Generación de un schedule empleando el algoritmo G&T



Fuente: Autor

El PSO propuesto (MOPSO) por Sha y Lin (2010), difiere del PSO original en la información almacenada en las soluciones pbest y gbest. Mientras que el PSO original, mantiene las mejores posiciones encontradas (x^k) hasta ahora, MOPSO mantiene el mejor schedule (S^k) generado por el algoritmo de G&T.

Se tiene el schedule generado anteriormente por el algoritmo G&T;

$$S = \{(1,1), (3,2), (3,1), (1,2), (3,3), (2,1), (1,3), (2,3), (2,2)\}$$

Se asignan preferencias teniendo en cuenta las máquinas y el orden en el que se encuentra cada operación en el conjunto S, construyendo así la matriz S^k:

- Para la máquina **1**

  $S = \{(\mathbf{1,1}), (3,2), (\mathbf{3,1}), (1,2), (3,3), (\mathbf{2,1}), (1,3), (2,3), (2,2)\}$, y la secuencia de operaciones en ella; (1,1) → (3,1) → (2,1), con prioridades 1, 2 y 3, respectivamente.

$$S^k = \begin{matrix} \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_3} \\ \begin{bmatrix} 1 & 3 & 2 \\ & & \end{bmatrix} & \begin{matrix} \boldsymbol{m_1} \\ \boldsymbol{m_2} \\ \boldsymbol{m_3} \end{matrix} \end{matrix}$$

- Para la máquina **2**

  $S = \{(1,1), (\mathbf{3,2}), (3,1), (\mathbf{1,2}), (3,3), (2,1), (1,3), (2,3), (\mathbf{2,2})\}$, la secuencia de operaciones es; (3,2) → (1,2) → (2,2), con prioridades 1, 2 y 3 respectivamente.

$$S^k = \begin{matrix} \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_3} \\ \begin{bmatrix} 1 & 3 & 2 \\ 2 & 3 & 1 \\ & & \end{bmatrix} & \begin{matrix} \boldsymbol{m_1} \\ \boldsymbol{m_2} \\ \boldsymbol{m_3} \end{matrix} \end{matrix}$$

- Para la máquina **3**

  $S = \{(1,1), (3,2), (3,1), (1,2), (\mathbf{3,3}), (2,1), (\mathbf{1,3}), (\mathbf{2,3}), (2,2)\}$, la secuencia de operaciones es; (3,3) → (1,3) → (2,3), con prioridades 1, 2 y 3 respectivamente.

$$S^k = \begin{matrix} \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_3} \\ \begin{bmatrix} 1 & 3 & 2 \\ 2 & 3 & 1 \\ 2 & 3 & 1 \end{bmatrix} & \begin{matrix} \boldsymbol{m_1} \\ \boldsymbol{m_2} \\ \boldsymbol{m_3} \end{matrix} \end{matrix}$$

Teniendo de esta forma la matriz S^k $(pbest)$ completa.

5.2 VELOCIDAD DE LA PARTÍCULA

El concepto de la velocidad original de PSO asume que cada partícula se mueve de acuerdo con la velocidad determinada por la distancia entre la posición anterior de la partícula y la solución gbest (pbest). Los dos propósitos principales de la velocidad de las partículas son de mantener la partícula que se mueve hacia las soluciones gbest y pbest, y mantener la inercia para evitar que las partículas queden atrapadas en óptimos locales. En MOPSO, los autores se concentran en prevenir que las partículas queden atrapadas en óptimos locales, en lugar de moverlas hacia la solución gbest (pbest).

Para un problema nxm, la velocidad de la partícula k se puede representar como una matriz mxn:

$$v^k = \begin{bmatrix} v_{11}^k & v_{21}^k & \cdots & v_{n1}^k \\ v_{12}^k & v_{22}^k & \cdots & v_{n2}^k \\ \vdots & \vdots & & \vdots \\ v_{1m}^k & v_{2m}^k & \cdots & v_{nm}^k \end{bmatrix} \qquad v_{ij}^k \in \{-1, 0, 1\}$$

Donde v_{ij}^k es la velocidad de la operación O_{ij}^k de la particula k.

El peso inercial W influye en la velocidad de las partículas en el PSO. La velocidad inicial de las partículas se genera aleatoriamente al comienzo de la iteración. En lugar de considerar la distancia desde x_{ij}^k a $pbest_i^k$ ($gbest_i$), MOPSO considera si el valor de x_{ij}^k es mayor o menor que $pbest_i^k$ ($gbest_i$). Si x_{ij}^k disminuye en la presente iteración, significa que $pbest_i^k$ ($gbest_i$) es más pequeño que x_{ij}^k, entonces x_{ij}^k se establece en movimiento hacia $pbest_i^k$ ($gbest_i$) dejando $v_{ij}^k \leftarrow -1$. Por lo tanto, en la siguiente iteración, $x_{ij}^k \leftarrow x_{ij}^k - 1$ con probabilidad W. Por el contrario, si x_{ij}^k

aumenta, entonces $pbest^k_i$ ($gbest_i$) es más grande que x_{ij}^k, y $v_{ij}^k \leftarrow 1$, por lo tanto, en la siguiente iteración, $x_{ij}^k \leftarrow x_{ij}^k + 1$ con probabilidad W.

Para cada partícula k y operación O_{ij} , si v_{ij}^k no es igual a cero (0), se establece en (0) con una probabilidad $(1 - W)$. Esto obliga a x_{ij}^k a detener el aumento o la disminución continua en esta iteración con probabilidad $(1 - W)$ mientras que x_{ij}^k se mantiene creciendo o disminuyendo. Entre mayor sea W, el valor de la prioridad seguirá creciendo o disminuyendo en más iteraciones y de esta manera es más difícil para la partícula devolverse a su posición anterior.

Una posible matriz de velocidades para el problema que se ha venido trabajando en este capítulo, podría ser la siguiente:

$$v^1 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

5.3 MOVIMIENTO DE LAS PARTÍCULAS

El movimiento de las partículas se basa en el operador de intercambio propuesto por Sha y Hsu (2006 y 2008), el procedimiento se produce tal como se muestra a continuación:

Paso 1: Se elige al azar una posición $\boldsymbol{\zeta}$ de x_{ij}^k.

Paso 2: Se marca el trabajo en la posición ζ de x_{ij}^k por Λ_1.

Paso 3: Si el número aleatorio rand $< C1$ entonces buscar la posición de Λ_1 en $pbest^k_i$; de lo contrario, buscar la posición de Λ_1 en $gbest_i$. Denotar la posición que

se ha encontrado en $pbest^k_i$ o $gbest_i$ por ζ', y el trabajo en la posición ζ' de x_{ij}^k por Λ_2. ($C1$ y $C2$ son constantes entre 0 y 1 tal que $C1 + C2 \leq 1$).

Paso 4: Si Λ_2 se ha indicado, $v_{ij1}^k = 0$ y $v_{ij2}^k = 0$, intercambiar Λ_1 y Λ_2 en x_{ij}^k, $v_{ij1}^k \leftarrow 1$

Paso 5: Si se han considerado todas las posiciones de x_{ij}^k, entonces se detiene. Si no, y si $\zeta < n$, entonces $\zeta \leftarrow \zeta + 1$, de lo contrario, $\zeta \leftarrow 1$. Volver al paso 2.

Se aplica el procedimiento de intercambio al problema que se ha venido trabajando en este capítulo, ver **Anexo B**.

Una vez consideradas todas las posiciones y todas las máquinas, se obtiene la nueva matriz de posición y velocidad.

$$x^1 = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix} \qquad v^1 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

5.4 ESTRATEGIA DE DIVERSIFICACIÓN

Si todas las partículas tienen las mismas soluciones no dominadas, quedarán atrapadas en óptimos locales. Para evitar que esto suceda, Sha y Lin (2010) proponen una estrategia de diversificación para mantener las soluciones no dominadas diferentes. Una vez que cualquier nueva solución es generada, el conjunto solución no dominado se actualizará en tres situaciones:

1. Si la solución de la partícula $(pbest)$ domina la solución $gbest$, asignar la solución $pbest$ como la nueva $gbest$.

2. Si la solución de la partícula es igual a cualquier solución en el conjunto de soluciones no dominadas, reemplazar la solución no dominada por la solución de la partícula.

3. Si la solución de la partícula está dominada por la peor solución no dominada y es diferente a cualquier solución, establecer la peor solución no dominada igual a la solución de la partícula.

6 BIBLIOTECA DE "BENCHMARK PROBLEMS" PARA EL JSP

Los problemas de comparación (benchmark problems) se usan para encontrar las ventajas comparativas de los diferentes métodos. Los problemas de comparación se prueban en los mismos tipos de problemas y clases de instancias, con el fin de establecer un estándar común para que los algoritmos sobre el JSP se puedan comparar entre sí. Los benchmark problems son usados comúnmente para medir la efectividad de los diferentes métodos, estos actúan como una plataforma estándar para que los algoritmos puedan probarse y medirse.

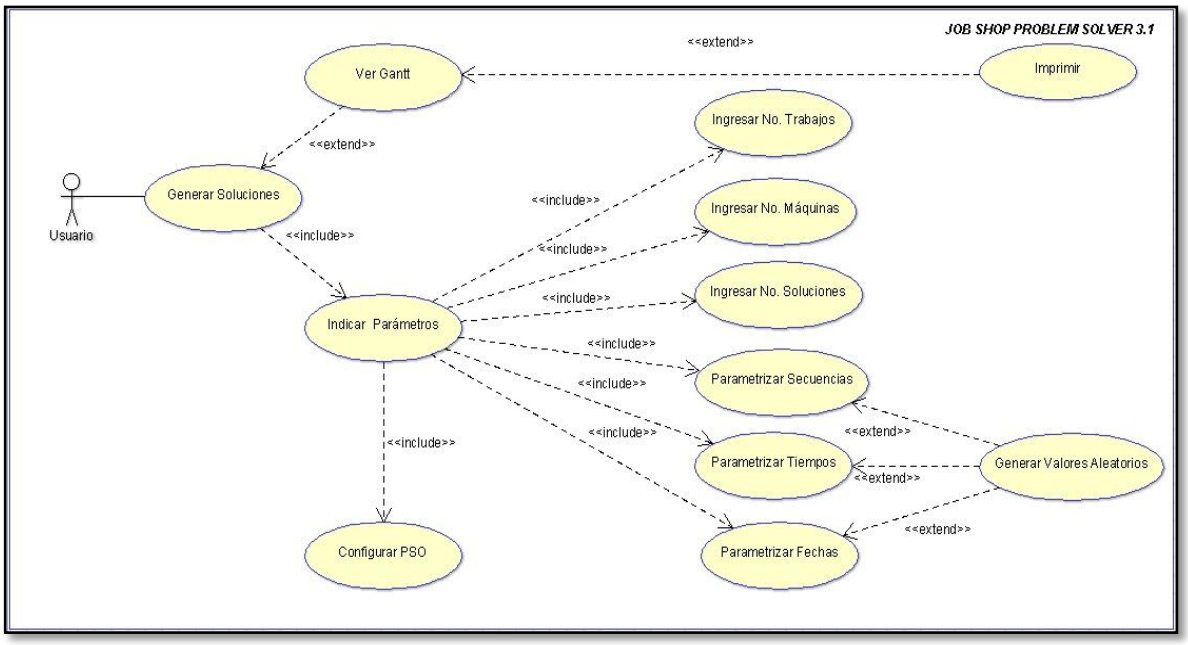
Estos problemas han sido propuestos por varios autores, entre los que se destacan, Fisher y Thompson (1963), Lawrence (1984), Adams et al. (1988), Applegate y Cook (1991), Storer et al. (1992) y Yamada y Nakano (1992), entre otros. Los dos problemas de benchmark más conocidos son los formulados por Muth y Thompson de tamaño 10x10 y 20x5 (mt10 y mt20 respectivamente) que son usados como "test beds " para medir la efectividad de cierto método. El problema mt10 permaneció sin resolver por más de 20 años aunque ya no es un reto computacional. Applegate y Cook proponen un conjunto de problemas de benchmark llamado los "10 problemas más difíciles" ("ten tough problems") como un reto computacional mayor que el mt10 de los cuales algunos no han sido resueltos.[37]

[37] SARMIENTO, C. Método Particle Swarm Optimization (PSO – Enjambre de Partículas) aplicado al problema de múltiples objetivos del Job Shop Scheduling (JSP) o secuenciamiento de máquinas. Escuela Ingeniería Industrial, Universidad Industrial de Santander, 2012.

7 HERRAMIENTA SOFTWARE

7.1 DIAGRAMA UML

Figura 17. Diagrama de casos de uso



Fuente: Autor

7.2 ENTORNO COMPUTACIONAL

El software se compilo para el sistema operativo Windows 7, el código fue desarrollado en lenguaje Java utilizando el entorno de desarrollo (IDE) Eclipse Versión Kepler Release y JDK 1.7 U 25. Las pruebas y ejecuciones del programa fueron realizadas en una computadora con procesador Intel Core i5 a 2.50 GHZ con 6 GB de RAM.

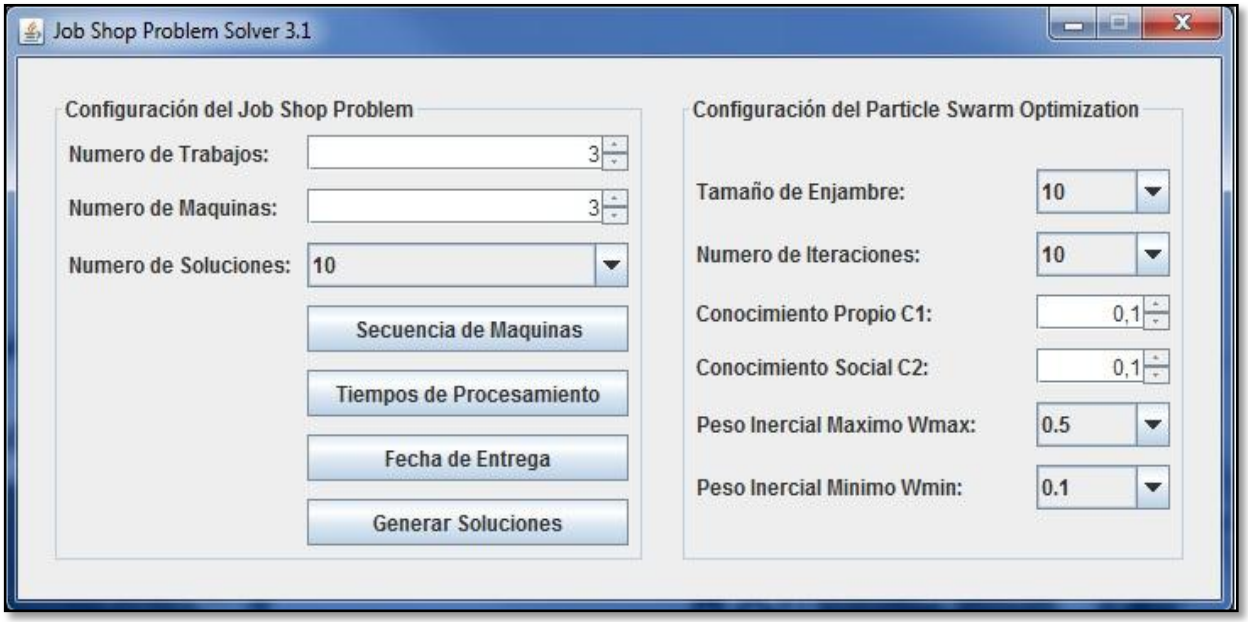
7.3 APLICACIÓN SOFTWARE

Para el desarrollo de la herramienta software se toma como base el diagrama de flujo MOPSO[38] del **Anexo C**.

7.3.1 EJECUCIÓN Y DATOS DE ENTRADA

Tras iniciar la aplicación se deben capturar los datos de entrada necesarios para definir la instancia o configuración del problema JSP y de la técnica PSO. A continuación la interfaz de captura.

Figura 18. Interfaz de inicio de la aplicación



Fuente: Autor

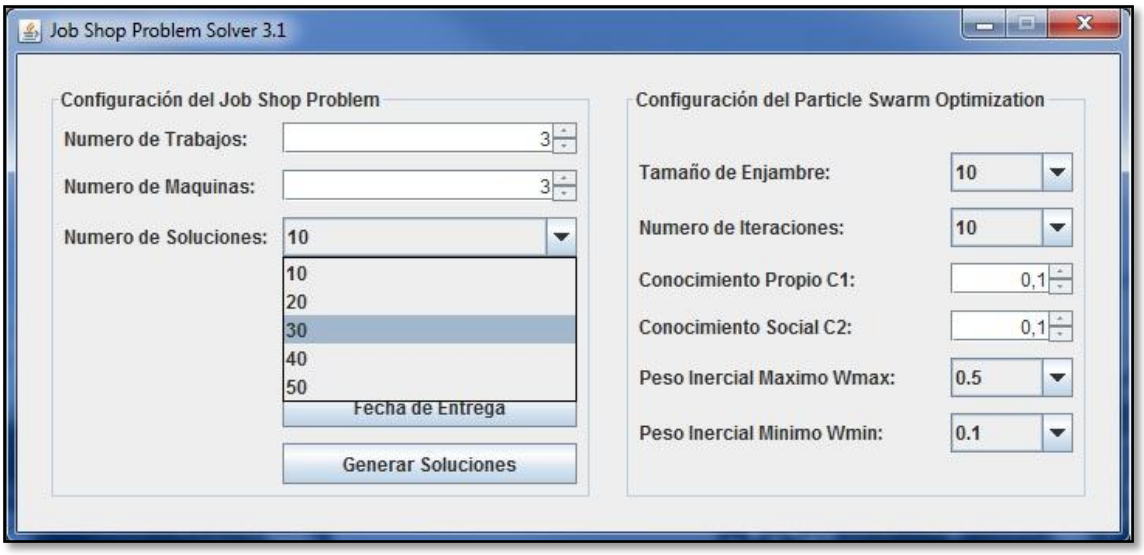
[38] SARMIENTO Ibid.

Tabla 5. Parámetros de JSP y PSO

PARAMETROS DE ENTRADA PARA JSP				
PARAMETRO	**TIPO DE CAPTURA**	**TIPO DE DATO**	**VALOR MINIMO**	**VALOR MAXIMO**
Número de trabajos	Spin Box	Entero	3 (default)	50
Número de máquinas	Spin Box	Entero	3 (default)	50
Número de soluciones	List Box	Entero	10 (default)	50
Secuencia de máquinas	Spin Box	Entero	3	50
Tiempos de procesamiento	Spin Box	Entero	1 (default)	100
Fecha de entrega	Spin Box	Entero	1 (default)	200
PARAMETROS DE ENTRADAS PARA PSO				
Tamaño de enjambre	List Box	Entero	10	50
Número de iteraciones	List Box	Entero	10	50
Conocimiento propio C1	Spin Box	Real	0,1	1
Conocimiento social C2	Spin Box	Real	0,1	1
Peso inercial máximo Wmax	List Box	Real	0,5 (default)/0,7	0,9
Peso inercial mínimo Wmin	List Box	Real	0,1 (default)/0,3	0,5

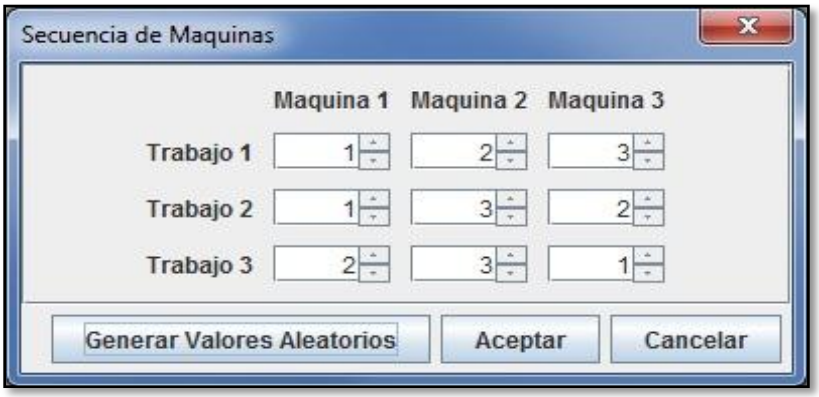
Fuente: Autor

Figura 19. Captura del número de soluciones



Fuente: Autor

Figura 20. Captura para secuencias de máquinas y tiempos de procesamiento



Fuente: Autor

Figura 21. Captura para fechas de entrega por trabajo

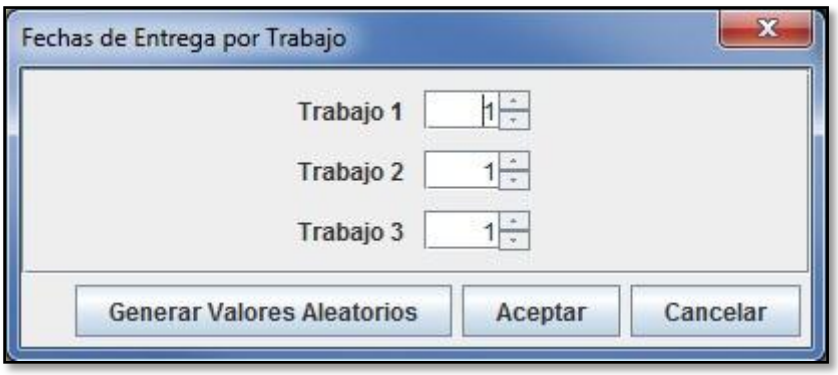
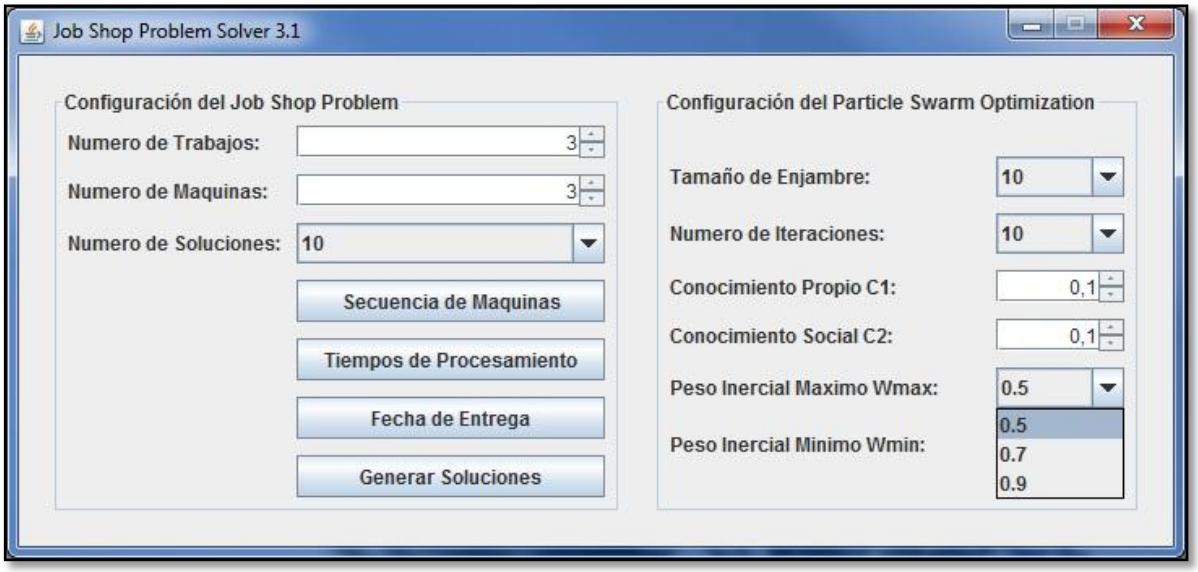


Figura 22. Captura de pesos inerciales



Fuente: Autor

7.3.2 VALIDACIÓN DE PARAMETROS DE ENTRADA

El usuario puede dejar los valores por defecto que se encuentran en los campos de captura, puede introducirlos o generarlos aleatoriamente. Una vez que se tengan los campos de los parámetros indicados se procede a dar clic en el

botón ***Generar Soluciones***. Dicho botón entonces validará que los campos se encuentren debidamente diligenciados, de lo contrario mostrará una ventana de alerta.

Tras indicarse los valores de los parámetros, el usuario debe aceptarlos, de lo contrario la aplicación lanza las siguientes alertas.

Figura 23. Validación de parámetros secuencia de máquinas, tiempos de procesamiento y fecha de entrega

Fuente: Autor

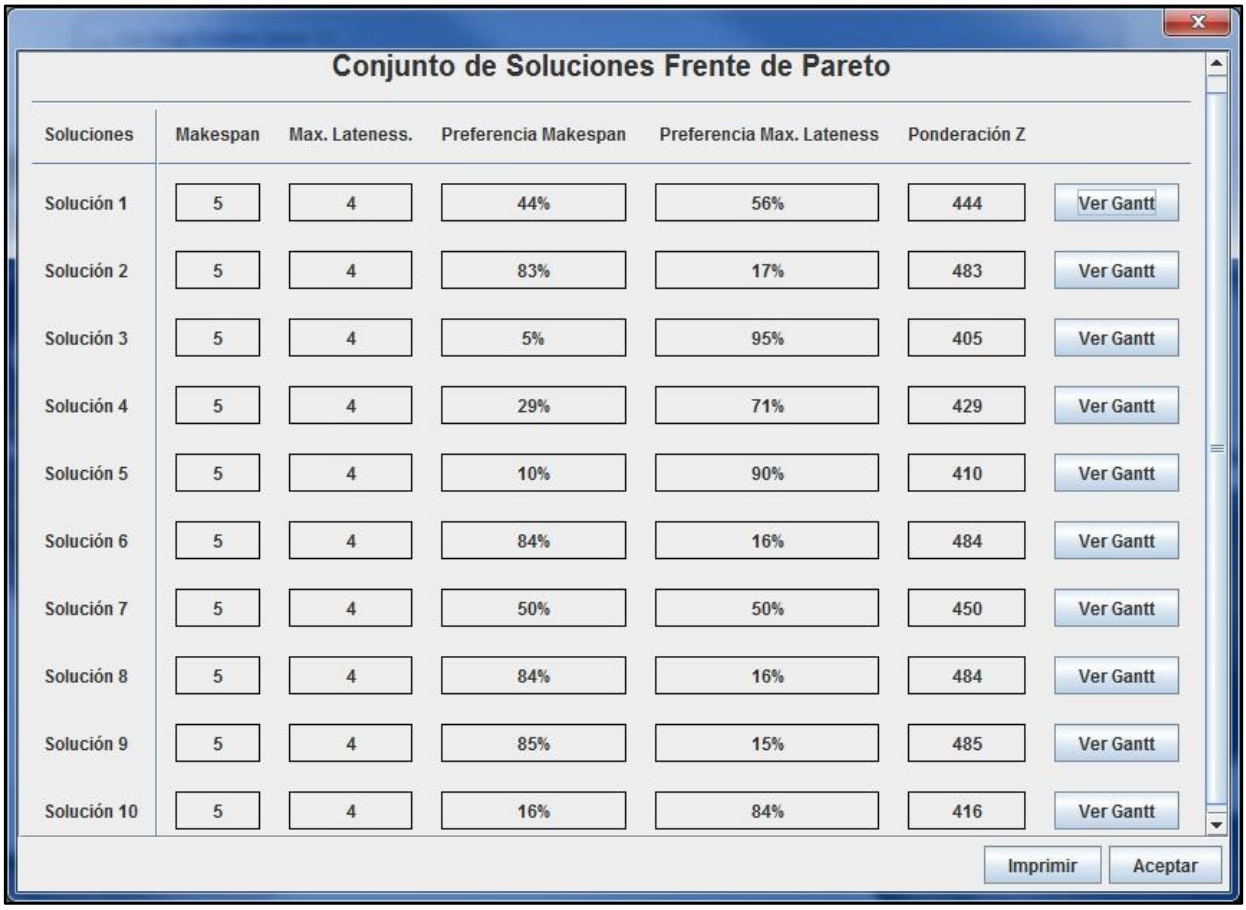
Se valida que la suma de los conocimientos C1, C2 sea menor o igual que uno ***(C1+C2 ≤ 1).***

Figura 24. Validación parámetros conocimiento propio y social

7.3.3 RESULTADOS

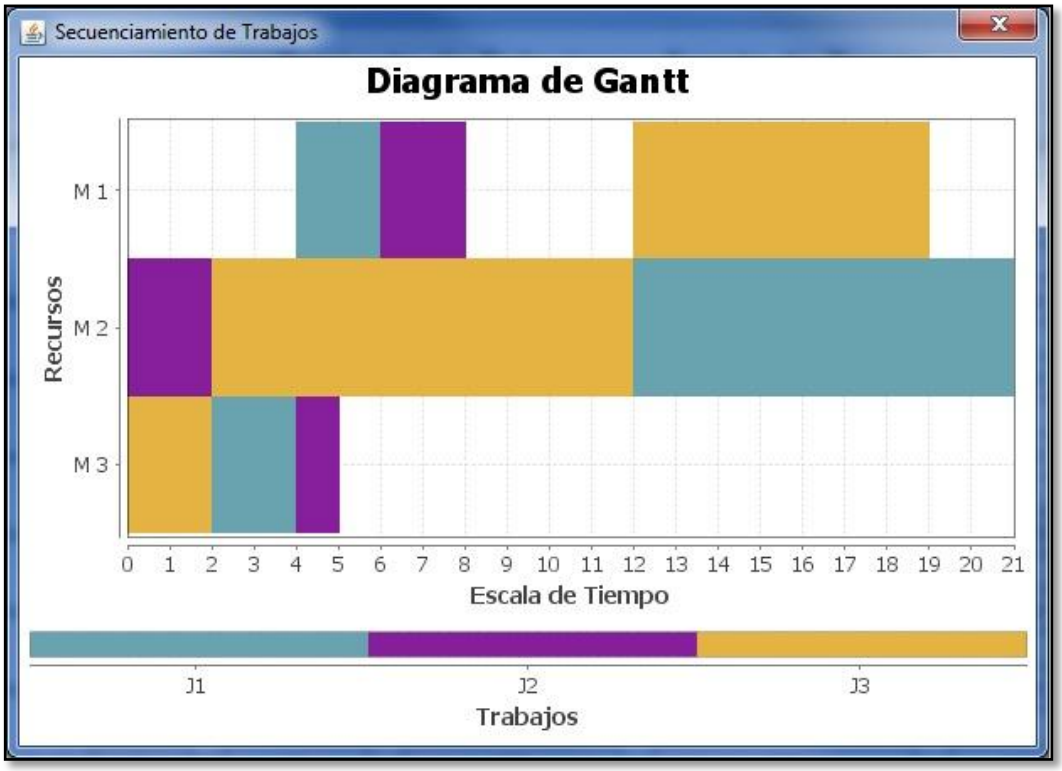
Tras haber configurado los parámetros del JSP y de PSO, el usuario debe **Generar Soluciones**, seguido se muestran los resultados. Se tabulan las soluciones correspondientes al Conjunto de Soluciones del Frente de Pareto, para cada una de ellas se asocia su respectivo diagrama de Gantt que es observable por el usuario cuando se da click sobre el botón **Ver Gantt**.

Figura 25. Soluciones Conjunto Frente de Pareto



Fuente: Autor

Figura 26. Diagrama de Gantt



Fuente: Autor

7.4 PRUEBA DE LA APLICACIÓN PARA UNA INSTANCIA 5X5

Se procede a indicar los valores de los parámetros de entrada.

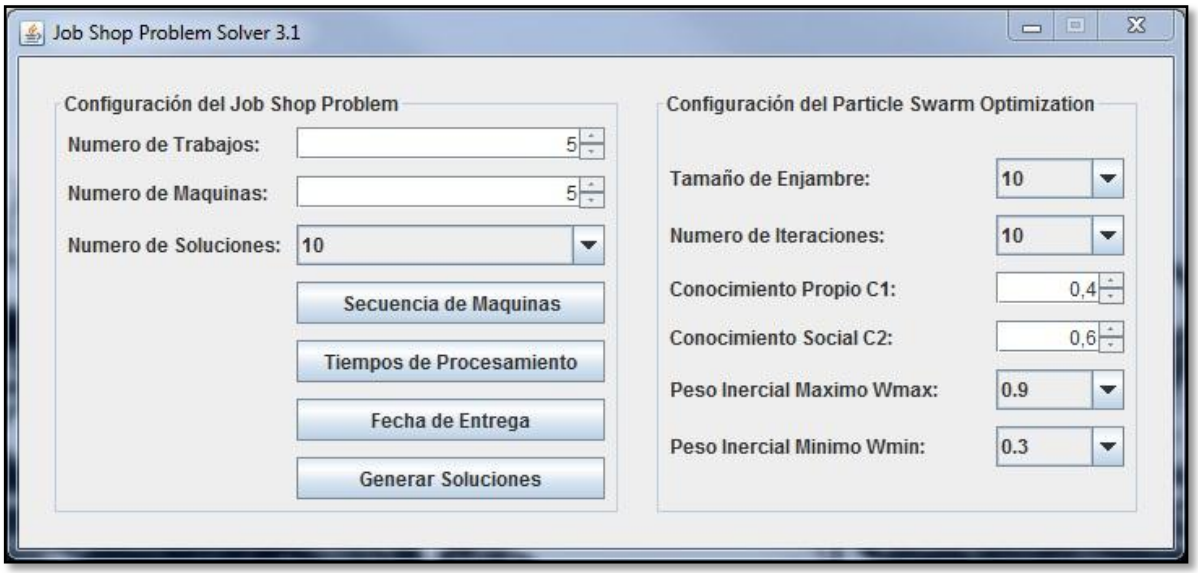
Tabla 6. Configuración de una instancia 5X5

PARAMETRO DE ENTRADA PARA JSP		PARAMETRO DE PSO	
PARAMETRO	**VALOR**	**PARAMETRO**	**VALOR**
Número de trabajos	5	Tamaño de enjambre	10
Número de máquinas	5	Número de iteraciones	10
Número de soluciones	10	Conocimiento propio C1	0,4
Secuencia de	Valores generados	Conocimiento	0,6

máquinas	aleatoriamente	social C2	
Tiempos de procesamiento	Valores generados aleatoriamente	Peso inercial máximo Wmax	0,9
Fecha de entrega	Valores generados aleatoriamente	Peso inercial mínimo Wmin	0,3

Fuente: Autor

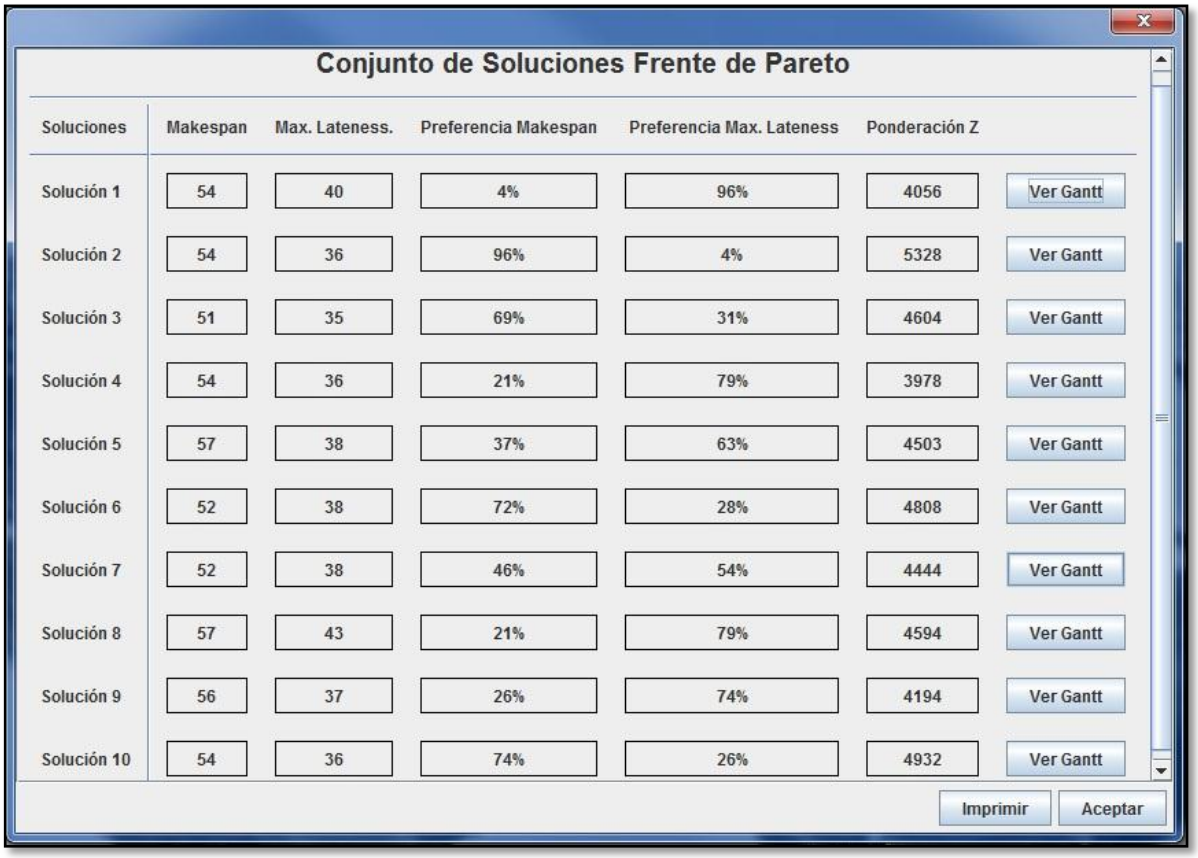
Figura 27. Configuración de una instancia 5X5



Fuente: Autor

Se indica ***Generar Soluciones*** y se obtiene 10 soluciones no comparables entre sí del Conjunto Eficiente de Pareto, para cada una de ellas se indica el valor de las funciones objetivo ***Makespan y Maximum Lateness***, sus respectivas preferencias y valor de ponderación Z. Para aproximar el conjunto solución se utiliza la técnica del método de las ponderaciones con coeficientes de ponderación enteros entre 0 y 100 generados aleatoriamente, un peso inercial linealmente decreciente y los valores de los parámetros definidos en la ***Tabla 6.***

Figura 28. Solución para una instancia 5X5



Fuente: Autor

7.5 JOB SHOP PROBLEM SOLVER 3.1 VS BENCHMARK

La herramienta software fue probada para el problema ***la05*** de la biblioteca de benchmark, cuya instancia se muestra a continuación:

```
# JSS
# 10X5 intervalo makespan (593, 677);

J1-  1 72  0 87  4 95  2 66  3 60
J2-  4 5   3 35  0 48  2 39  1 54
J3-  1 46  3 20  2 21  0 97  4 55
J4-  0 59  3 19  4 46  1 34  2 37
J5-  4 23  2 73  3 25  1 24  0 28
J6-  3 28  0 45  4 5   1 78  2 83
J7-  0 53  3 71  1 37  4 29  2 12
J8-  4 12  2 87  3 33  1 55  0 38
J9-  2 49  3 83  1 40  0 48  4 7
J10- 2 65  3 17  0 90  4 27  1 23
```

Lo anterior equivale a lo siguiente:

Tabla 7. Tiempos de procesamiento benchmark la05

tiempos de procesamiento														
	m	**t**		**m**	**t**		**m**	**t**		**m**	**t**		**m**	**t**
J1	1	87	**J2**	1	48	**J3**	1	97	**J4**	1	59	**J5**	1	28
	2	72		2	54		2	46		2	34		2	24
	3	66		3	39		3	21		3	37		3	73
	4	60		4	35		4	20		4	19		4	25
	5	95		5	5		5	55		5	46		5	23
J6	**m**	**t**	**J7**	**m**	**t**	**J8**	**m**	**t**	**J9**	**m**	**t**	**J10**	**m**	**t**
	1	45		1	53		1	38		1	48		1	90
	2	78		4	71		2	55		2	40		2	23
	3	83		2	37		3	87		3	49		3	65
	4	28		5	29		4	33		4	83		4	17
	5	5		3	12		5	12		5	7		5	27

Fuente: Autor

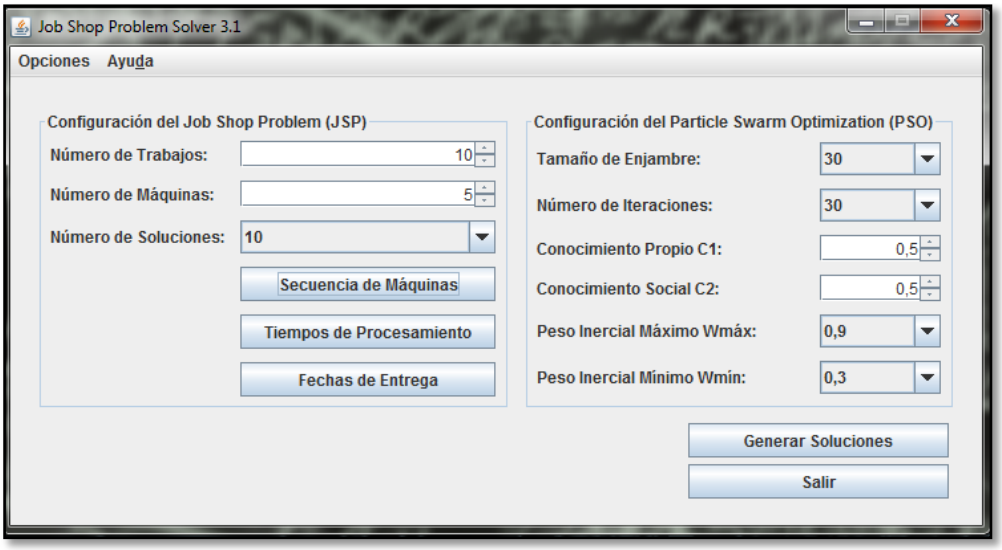
Tabla 8. Secuenciamiento benchmark la05

	secuenciamiento				
Trabajo	**máquinas**				
1	2	1	5	3	4
2	5	4	1	3	2
3	2	4	3	1	5
4	1	4	5	2	3
5	5	3	4	2	1
6	4	1	5	2	3
7	1	4	2	5	3
8	5	3	4	2	1
9	3	4	2	1	5
10	3	4	1	5	2

Fuente: Autor

A continuación se muestra el resultado de la configuración de los parámetros de PSO para el problema benchmark **la05** en la aplicación software.

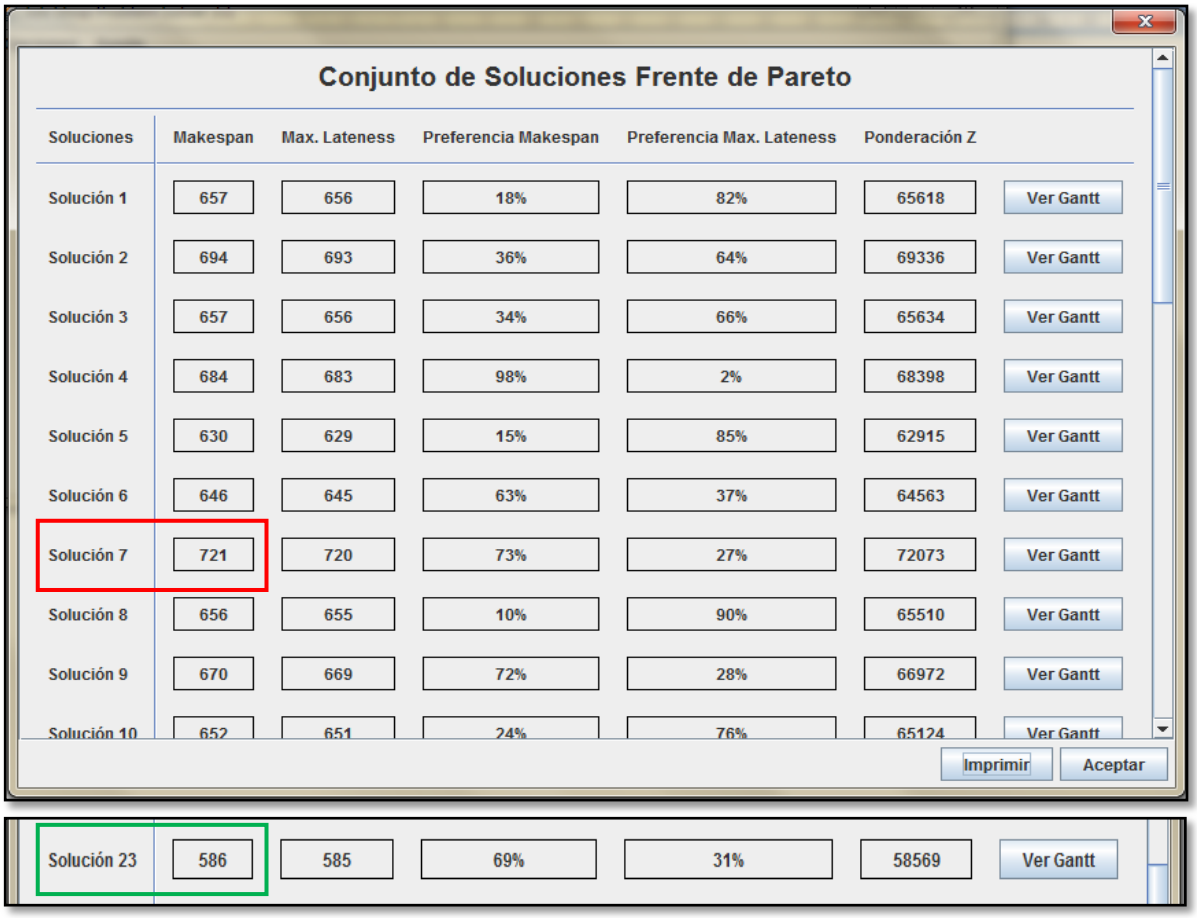
Figura 29. Configuración de PSO para el benchmark la05



Fuente: Autor

La solución obtenida se muestra en la siguiente figura:

Figura 30. Solución al benchmark la05



Fuente: Autor

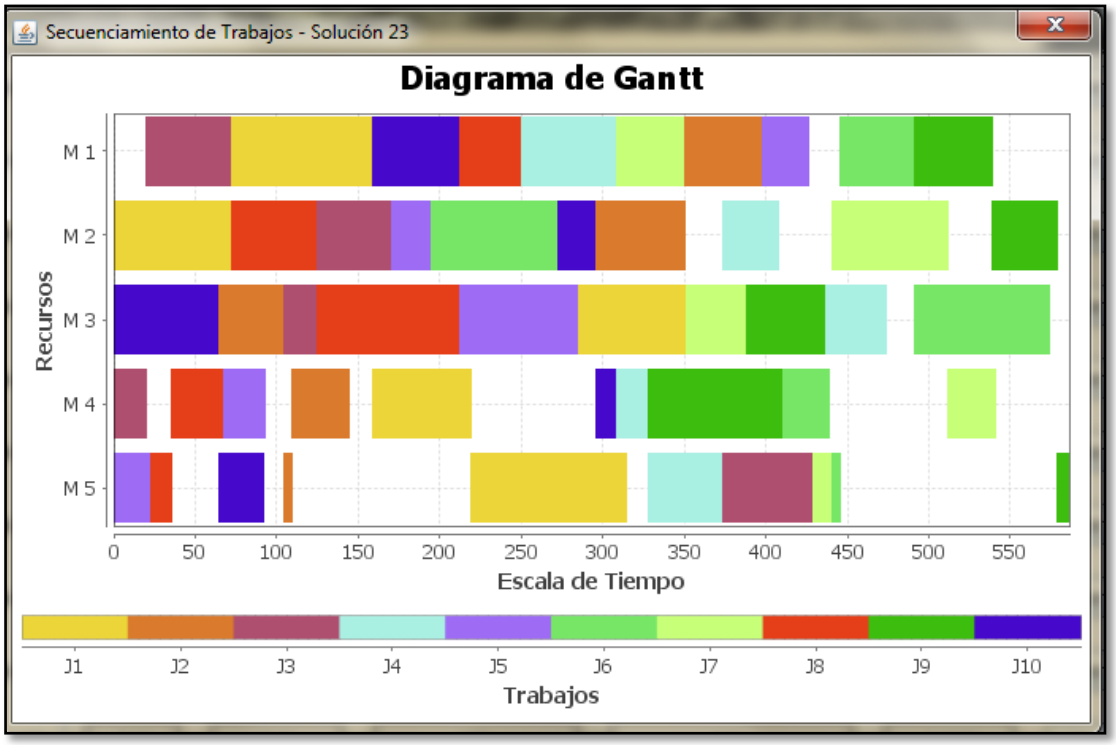
Como se puede observar en la figura 30, se obtiene el mejor valor del objetivo makespan en la solución 23 con 586 unidades de tiempo, frente a un valor de 593 reportado en la literatura[39], indicando así, que la aplicación desarrollada junto con los algoritmos implementados en ella proporcionan buenas o mejores soluciones que las ya encontradas. Se contrasta en cambio en el valor de la peor solución

[39] http://www.eii.uva.es/elena/JSSP/InstancesJSSP.htm

que para el caso corresponde a la solución 7 con un makespan de 721 unidades de tiempo, contra 677 reportadas para el mismo problema.

En la siguiente figura se muestra el diagrama de Gantt para la mejor solución encontrada por Job Shop Problem Solver 3.1.

Figura 31. Mejor solución para benchmark la05



Fuente: Autor

8 CONCLUSIONES

- Se logró desarrollar una aplicación que permite dar solución simultánea a dos objetivos; makespan y maximum lateness, aproximando sus soluciones al Conjunto Frente de Pareto para instancias nxm con valores entre 3 y 30 para el número de trabajos y para el número de máquinas. Esto se logró implementando el pseudocódigo propuesto en (SARMIENTO, 2012).

- La aplicación software permite dar rápida respuesta a problemas del mundo real del tipo JSP, sirviendo así como soporte operativo en la planeación de las actividades y en la toma de decisiones propias de la industria manufacturera.

- El tiempo de respuesta de la aplicación para generar las soluciones del Conjunto Frente de Pareto aumenta proporcionalmente con el tamaño del problema, es decir, cuando las variables de entrada número de trabajos, número de maquinas y número de soluciones aumentan.

- El valor de las variables de entrada para la metaheurística Particle Swarm Optimization influye significativamente en los tiempos de respuesta de la aplicación, cuanto mayor sean sus valores, más tiempo toma la aplicación en generar la solución del conjunto Frente de Pareto.

9 RECOMENDACIONES

- Se recomienda desarrollar una nueva versión de la aplicación Job Shop Problem Solver 3.1, donde el usuario disponga de un listado de funciones objetivo a optimizar y donde según las necesidades él pueda seleccionar cuales quiera de ellas.

- Debido a que el coste computacional aumenta en la medida que parámetros como: número de trabajos, número de máquinas, número de iteraciones, tamaño del enjambre y el numero de soluciones aumentan, se recomienda abordar este tipo de problemas en arquitecturas computacionales tipo clúster, ya que de esta forma se logra reducir el tiempo de cómputo.

- Se recomienda fortalecer los lazos de cooperación entre las escuelas de ingeniería de sistemas e industrial, con el fin de investigar e implementar otros métodos (algoritmos genéticos, búsqueda tabú, recocido simulado, entre otros) de solución aplicables a este y otros tipos de problemas de la industria manufacturera, tales como Flow Shop, el Open Shop y el Flexible Open/Flow Shop, etc. Aplicado, tanto en ámbitos de múltiples objetivos como de un solo objetivo.

BIBLIOGRAFÍA

BALAS, E. (1969). Machine Sequencing via Disjunctive Graphs: an Implicit Enumeration Algorithm. *Operations Research* , vol. 17, pag. 941-957.

BOOKER, L. (Los Altos, CA. 1987). Improving Search in Genetic Algorithms . *Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publishers* , pag. 61-73.

CAGNINA, L. (2010). *Optimización Mono y Multiobjetivo a través de una Heurística de Inteligencia Colectiva.* Argentina: Doctorado en Ciencias de la Computación, Universidad Nacional de San Luis.

CENTENO, A. A. (Abril 2009). Problemas de Optimización en Ingeniería y el Algoritmo de Enjambre de Partículas Departamento de Física, Facultad de Ingeniería, Universidad de Carabobo, Valencia, Venezuela. *Revista Ingeniería UC* , Vol . 16, Nº. 1, pag. 49-54.

CORREA, R. B. (2010). *Validación de un algoritmo híbrido del PSO con el método simplex y de topología de evolución paramétrica. .* UIS.

DIAZ, A. L., & CADENA GONZÁLEZ, R. A. (2012). *Solución del problema de ruteo de vehículos con ventanas de tiempo (VRPTW) mediante métodos heurísticos.* Bucaramanga: Universidad Industrial de Santander.

DIAZ, N. L. (2012). *Implementación de un algoritmo immune artificial aplicado en el area de planificación de recursos.* Bucaramanga: Universidad industrial de Santander.

DURAN MEDINA ROSA, L. P. (2011). Un algoritmo genético para el problema de Job Shop Flexible. *Revista chilena de ingeniería , 19* (1), 53-61.

FRENCH, S. (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop.* Horwood, Chichester.

GIFFLER, B. y. (1960). Algorithms for Solving Production Scheduling Problems. *Operations Research* , vol. 8. nro. 4, pag. 487-503.

JAIN, A. y. (1999). Deterministic Job Shop Scheduling: Past, Present and Future. . *European Journal Of Operational Research* , Vol. 113, pp. 390-434.

KENNEDY, J. E. (1995). Particle Swarm Optimization. *Proc. IEEE Int. Conf. Neural Networks* , 39-43.

LATORRE, G. (Junio 2011). *El problema flow shop flexible de dos etapas: programación de las intervenciones quirúrgicas en un hospital .* universidad del bío-bío,facultad de ingeniería.

MARTÍ, R. *Procedimientos Metaheurísticos en Optimización Combinatoria.* Universitad de Valéncia: Departamento de Estadística e Investigación Operativa. Facultad de Matemáticas.

PEÑA, V. Z. (Mayo 2006). *Estado del Arte del Job Shop Scheduling Problem. Universidad Técnica Federico Santa María Valparaíso.* Chile.

PERÉZ, J. (2005). *Contribución a los metodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo.* Santander, España.

PINEDO, M. (1995). *Scheduling: Theory, Algorithms and Systemsl.* Prentice Hal.

ROMERO, C. (1996). *Análisis de las decisiones multicriterio.* Madrid: Isdefe.

ROY, B. y. (1964). *Les Problems d'Ordonnancemente avec Constintes Disjonctives.* Paris: Technical Report 9, SEMA, Note D.S.

SALTO, C. (2000). *Algoritmos evolutivos avanzados como soporte del proceso productivo.*

SARMIENTO, C. J. (2012). *Método Particle Swarm Optimization (PSO - Enjambre de Partículas) aplicado al problema de múltiples objetivos Job Shop Scheduling (JSP) o secuenciamiento de máquinas.* Bucaramanga.

SHA, D. Y. (2010). A multi-objective PSO for Job Shop-Scheduling Problems. *Expert Systems with Applications* , Vol. 37, pp. 1065-1070.

TAHA, H. (2004). *Investigación de Operaciones.* Mexico: Pearson.

TAMAKI, H. M. (1995). Generation of a Set of Pareto-Optimal Solutions by Genetic Algorithms. *Transactions of the Society of Instrument and Control Engineers* , vol. 31, nro. 8, pag.1185-1192.

XU, S. R. (Mar., 2007). Boundary conditions in particle swarm optimization revisited. *IEEE Trans. Antennas Propagat.* , Vol. 55, no.3, 760-765.

Y-L. Zheng, L.-H. M.-Y.-X. (Xi'an (China), Noviembre 2003). On the convergence analysis and parameter selection in particle swarm optimization. *Proceedings of*

the 2003 International Conference on Machine Learning and Cybernetics , Vol. 3, pp. 1802-1807.

ZHENG, Y. M. (2003). On the convergence analysis and parameter selection in particle swarm optimization. *Proceedings of the International Conference on Machine Learning and Cybernetics* , Vol. 3, pp. 1802-1807.

ANEXOS

Anexo A. Implementación del algoritmo de G&T en un problema 3x3

Las operaciones (i,j); trabajos $i = 1,2,3$, máquinas $j = 1,2,3$, que se usaran en el desarrollo del algoritmo teniendo en cuenta los datos mostrados en la **Tabla 5** son:

$$\begin{matrix} J_1 \\ J_2 \\ J_3 \end{matrix}\begin{Bmatrix} (1,1) & \rightarrow & (1,2) & \rightarrow & (1,3) \\ (2,1) & \rightarrow & (2,3) & \rightarrow & (2,2) \\ (3,2) & \rightarrow & (3,1) & \rightarrow & (3,3) \end{Bmatrix}$$

La posición de la partícula $k = 1$ se representa por una matriz $m \text{ x } n$:

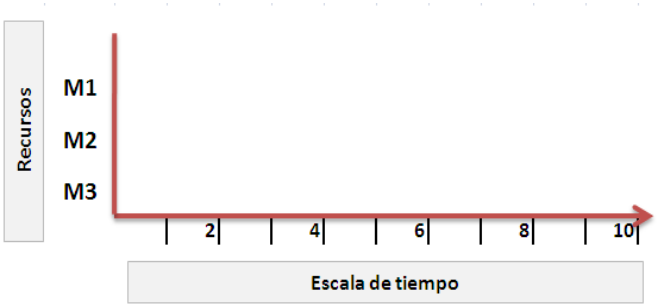
$$\boldsymbol{x^1} = \begin{matrix} \boldsymbol{J_1} & \boldsymbol{J_2} & \boldsymbol{J_3} \end{matrix}\begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix}\begin{matrix} \boldsymbol{m_1} \\ \boldsymbol{m_2} \\ \boldsymbol{m_3} \end{matrix}$$

Algoritmo G&T:

Inicialización:

Paso 1.

- Se inicializa el conjunto S que guarda el schedule $\rightarrow S = \emptyset$ (empieza vacio)



- Se asigna a Ω todas las operaciones sin predecesores

$$\boldsymbol{\Omega = \{(1,1),(2,1),(3,2)\}}$$

Iteración 1…

Paso 2:

- Se determina $f^* = min_{(i,j)\in\Omega}\{f_{(i,j)}\}$, $f_{(i,j)} = s_{(i,j)} + p_{(i,j)}$

  $s_{(1,1)} = 0, \quad s_{(2,1)} = 0, \quad s_{(3,2)} = 0$

  $p_{(1,1)} = 3, \quad p_{(2,1)} = 2, \quad p_{(3,2)} = 3$

  $f_{(1,1)} = 0 + 3 = 3, \quad f_{(2,1)} = 0 + 2 = 2, \quad f_{(3,2)} = 0 + 3 = 3$

  $f^* = min_{(i,j)\in\Omega}\{f_{(1,1)}, f_{(2,1)}, f_{(3,2)}\} \rightarrow f^* = min_{(i,j)\in\Omega}\{3,2,3\} \rightarrow f^* = f_{(2,1)} = 2$

- Se designa como m^* a la máquina en la que se realiza $f^* \rightarrow m^* = 1$

Paso 3:

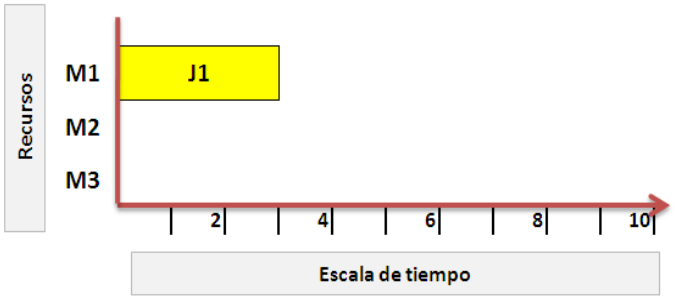
1) Se identifica el conjunto de operaciones $\in \Omega$ y requieran también la máquina m^* $(1) \rightarrow \{(1,1),(2,1)\}$.
Y que cumplan con $s_{(i,j)} < f^* \rightarrow s_{(i,j)} < 2$
$s_{(1,1)} = 0 < 2, \quad s_{(2,1)} = 0 < 2 \rightarrow \{(1,1),(2,1)\}$

2) Se elige la operación del conjunto de 3(1) con la mayor prioridad de acuerdo a la matriz x^1

- La operación $(1,1)$ tiene prioridad 1
- La operación $(2,1)$ tiene prioridad 3

Se escoge la operación $(1,1)$.

3) Añadir la operación anterior a $S \rightarrow S = \{(1,1)\}$



4) Asignar a $s_{(i,j)}$ el tiempo de finalización $f_{(1,1)} \rightarrow s_{(1,1)} = f_{(1,1)} = 3$, este tiempo de inicio se asigna a las siguientes operaciones que coincidan en la misma máquina o trabajo.

Paso 4: Como no se ha generado un schedule completo, se borra $(1,1)$ de Ω, y se incluye su sucesor inmediato de acuerdo con la matriz de operaciones a realizar.

$$\boldsymbol{\Omega = \{(1,2),(2,1),(3,2)\}}$$

Iteración 2…

Paso 2:

Se determina $f^* = min_{(i,j)\in\Omega}\{f_{(1,2)}, f_{(2,1)}, f_{(3,2)}\}$

$s_{(1,2)} = 3, \quad s_{(2,1)} = 3, \quad s_{(3,2)} = 0$

$p_{(1,2)} = 3, \quad p_{(2,1)} = 2, \quad p_{(3,2)} = 3$

$f_{(1,2)} = 3 + 3 = 6, \quad f_{(2,1)} = 3 + 2 = 5, \quad f_{(3,2)} = 0 + 3 = 3$

$f^* = min_{(i,j)\in\Omega}\{6,5,3\} \rightarrow f^* = f_{(3,2)} = 3$

$m^* = 2$

Paso 3:

1) Operaciones $\in \Omega$ y requieren $m^*(2) \rightarrow \{(1,2),(3,2)\}$.

Y que cumplan con $s_{(i,j)} < f^* \rightarrow s_{(i,j)} < 3$

$s_{(1,2)} = 3 < 3, \quad s_{(3,2)} = 0 < 3 \rightarrow \{(3,2)\}$

2) $\{(3,2)\}$

3) Añadir la operación $(3,2)$ a $S \rightarrow S = \{(1,1),(3,2)\}$

4) $s_{(3,2)} = f_{(3,2)} = 3$

Paso 4: Como no se ha generado un schedule completo, se borra $(3,2)$ de Ω, y se incluye su sucesor inmediato de acuerdo con la matriz de operaciones a realizar.

$$\boldsymbol{\Omega = \{(1,2),(2,1),(3,1)\}}$$

Iteración 3…

Paso 2:

- Se determina $\boldsymbol{f^* = min_{(i,j)\in\Omega}\{f_{(1,2)}, f_{(2,1)}, f_{(3,1)}\}}$

$\boldsymbol{s_{(1,2)} = 3, \quad s_{(2,1)} = 3, \quad s_{(3,1)} = 3}$

$$p_{(1,2)} = 3, \quad p_{(2,1)} = 2, \quad p_{(3,1)} = 3$$

$$f_{(1,2)} = 3 + 3 = 6, \quad f_{(2,1)} = 3 + 2 = 5, \quad f_{(3,1)} = 3 + 3 = 6$$

$$f^* = min_{(i,j)\in\Omega}\{6, 5, 6\} \rightarrow f^* = f_{(2,1)} = 5$$

- $m^* = 1$

Paso 3:

1) Operaciones $\in \Omega$ y requieren m^* (1) $\rightarrow \{(2,1),(3,1)\}$.

Y que cumplan con $s_{(i,j)} < f^* \rightarrow s_{(i,j)} < 5$

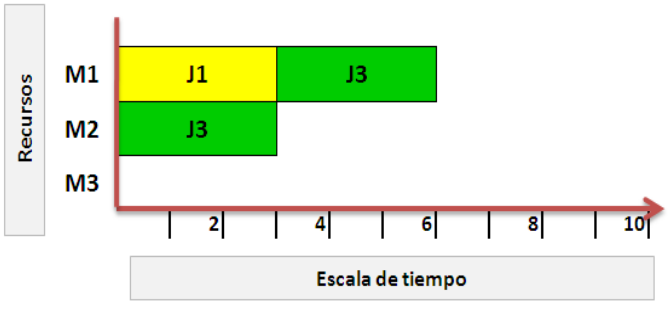
$s_{(2,1)} = 3 < 5, \quad s_{(3,1)} = 3 < 5 \rightarrow \{(2,1),(3,1)\}$

2) La operación (2,1) tiene prioridad 3

La operación (3,1) tiene prioridad 2

Se escoge $\{(3,1)\}$

3) Añadir la operación $(3,1)$ a $S \rightarrow S = \{(1,1),(3,2),(3,1)\}$



4) $s_{(3,1)} = f_{(3,1)} = 6$

Paso 4: Como no se ha generado un schedule completo, se borra $(\mathbf{3},\mathbf{1})$ de $\boldsymbol{\Omega}$, y se incluye su sucesor inmediato de acuerdo con la matriz de operaciones a realizar.

$$\boldsymbol{\Omega} = \{(\mathbf{1},\mathbf{2}),(\mathbf{2},\mathbf{1}),(\mathbf{3},\mathbf{3})\}$$

Iteración 4…

Paso 2:

- Se determina $\boldsymbol{f^* = min_{(i,j)\in\Omega}\{f_{(1,2)}, f_{(2,1)}, f_{(3,3)}\}}$

  $\boldsymbol{s_{(1,2)} = 3,} \quad \boldsymbol{s_{(2,1)} = 6,} \quad \boldsymbol{s_{(3,3)} = 6}$

  $\boldsymbol{p_{(1,2)} = 3,} \quad \boldsymbol{p_{(2,1)} = 2,} \quad \boldsymbol{p_{(3,3)} = 1}$

  $\boldsymbol{f_{(1,2)} = 3+3 = 6,} \quad \boldsymbol{f_{(2,1)} = 6+2 = 8,} \quad \boldsymbol{f_{(3,3)} = 6+1 = 7}$

  $\boldsymbol{f^* = min_{(i,j)\in\Omega}\{6,8,7\} \rightarrow f^* = f_{(1,2)} = 6}$

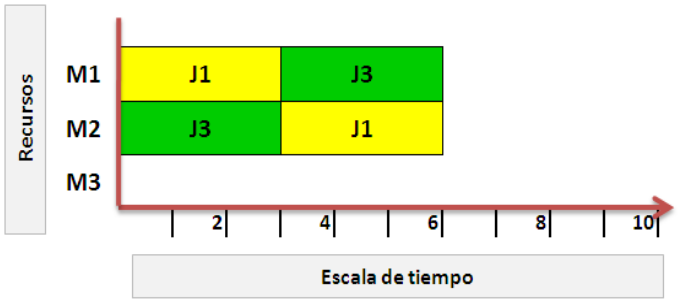
- $\boldsymbol{m^* = 2}$

Paso 3:

1) Operaciones $\in$ $\boldsymbol{\Omega}$ y requieren $\boldsymbol{m^*}$ (2) $\rightarrow \{(\mathbf{1},\mathbf{2})\}$.

2) $\{(\mathbf{1},\mathbf{2})\}$

3) Añadir $(\mathbf{1},\mathbf{2})$ a $\boldsymbol{S} \rightarrow \boldsymbol{S} = \{(\mathbf{1},\mathbf{1}),(\mathbf{3},\mathbf{2}),(\mathbf{3},\mathbf{1}),(\mathbf{1},\mathbf{2})\}$



4) $s_{(1,2)} = f_{(1,2)} = 6$

Paso 4: Como no se ha generado un schedule completo, se borra $(1, 2)$ de Ω, y se incluye su sucesor inmediato de acuerdo con la matriz de operaciones a realizar.

$$\Omega = \{(1, 3), (2, 1), (3, 3)\}$$

Iteración 5…

Paso 2:

- Se determina $f^* = min_{(i,j)\in\Omega}\{f_{(1,3)}, f_{(2,1)}, f_{(3,3)}\}$

  $s_{(1,3)} = 6, \quad s_{(2,1)} = 6, \quad s_{(3,3)} = 6$

  $p_{(1,3)} = 3, \quad p_{(2,1)} = 2, \quad p_{(3,3)} = 1$

  $f_{(1,3)} = 6 + 3 = 9, \quad f_{(2,1)} = 6 + 2 = 8, \quad f_{(3,3)} = 6 + 1 = 7$

  $f^* = min_{(i,j)\in\Omega}\{9, 8, 7\} \rightarrow f^* = f_{(3,3)} = 7$

- $m^* = 3$

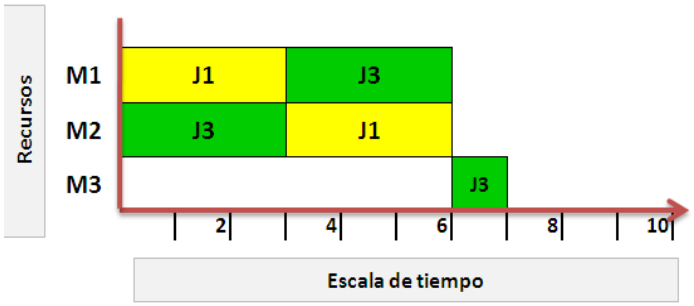
Paso 3:

1) Operaciones ∈ $\boldsymbol{\Omega}$ y requieren $\boldsymbol{m}^*$ (3) → $\{(\mathbf{1},\mathbf{3}),(\mathbf{3},\mathbf{3})\}$.

$s_{(1,3)} = 6 < 7, \quad s_{(3,3)} = 6 < 7 \rightarrow \{(1,3),(3,3)\}$

2) $\{(\mathbf{3},\mathbf{3})\}$

3) Añadir $(\mathbf{3},\mathbf{3})$ a $\boldsymbol{S} \rightarrow \boldsymbol{S} = \{(\mathbf{1},\mathbf{1}),(\mathbf{3},\mathbf{2}),(\mathbf{3},\mathbf{1}),(\mathbf{1},\mathbf{2}),(\mathbf{3},\mathbf{3})\}$



4) $\boldsymbol{s}_{(3,3)} = \boldsymbol{f}_{(3,3)} = \mathbf{7}$

Paso 4: Como no se ha generado un schedule completo, se borra $(\mathbf{3},\mathbf{3})$ de $\boldsymbol{\Omega}$.

$$\boldsymbol{\Omega} = \{(\mathbf{2},\mathbf{1}),(\mathbf{1},\mathbf{3})\}$$

Iteración 6…

Paso 2:

- Se determina $\boldsymbol{f}^* = \boldsymbol{min}_{(i,j)\in\boldsymbol{\Omega}}\{\boldsymbol{f}_{(1,3)}, \boldsymbol{f}_{(2,1)}\}$

$\boldsymbol{s}_{(1,3)} = \mathbf{7}, \qquad \boldsymbol{s}_{(2,1)} = \mathbf{6}$

$$p_{(1,3)} = 3, \qquad p_{(2,1)} = 2$$

$$f_{(1,3)} = 7 + 3 = 10, \qquad f_{(2,1)} = 6 + 2 = 8$$

$$f^* = min_{(i,j)\in\Omega}\{10, 8\} \rightarrow f^* = f_{(2,1)} = 8$$

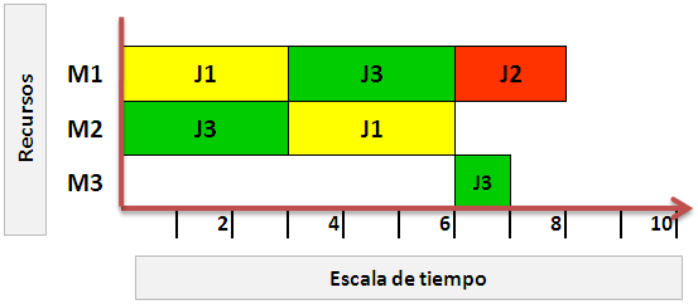
- $m^* = 1$

Paso 3:

1) Operaciones $\in \Omega$ y requieren m^* (1) $\rightarrow \{(2,1)\}$.

2) $\{(2,1)\}$

3) Añadir $(2,1)$ a $S \rightarrow S = \{(1,1),(3,2),(3,1),(1,2),(3,3),(2,1)\}$



4) $s_{(2,1)} = f_{(2,1)} = 8$

Paso 4: Como no se ha generado un schedule completo, se borra $(2,1)$ de Ω, y se incluye su sucesor inmediato.

$$\Omega = \{(2,3),(1,3)\}$$

Iteración 7…

Paso 2:

- Se determina $f^* = min_{(i,j)\in\Omega}\{f_{(1,3)}, f_{(2,3)}\}$

  $s_{(1,3)} = 7, \quad s_{(2,3)} = 8$

  $p_{(1,3)} = 3, \quad p_{(2,3)} = 3$

  $f_{(1,3)} = 7 + 3 = 10, \quad f_{(2,3)} = 8 + 3 = 11$

  $f^* = min_{(i,j)\in\Omega}\{10, 11\} \rightarrow f^* = f_{(1,3)} = 10$

- $m^* = 3$

Paso 3:

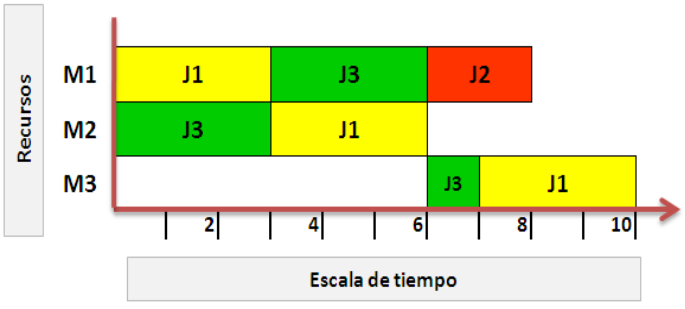
1) Operaciones $\in \Omega$ y requieren m^* (3) $\rightarrow \{(2,3),(1,3)\}$.
$s_{(2,3)} = 8 < 10, \quad s_{(1,3)} = 7 < 10 \rightarrow \{(2,3),(1,3)\}$

La operación $(2,3)$ tiene prioridad 3

La operación $(1,3)$ tiene prioridad 2

2) $\{(1,3)\}$

3) Añadir $(1,3)$ a $S \rightarrow S = \{(1,1),(3,2),(3,1),(1,2),(3,3),(2,1),(1,3)\}$



4) $\boldsymbol{s_{(1,3)} = f_{(1,3)} = 10}$

Paso 4: Como no se ha generado un schedule completo, se borra $(\mathbf{1},\mathbf{3})$ de $\boldsymbol{\Omega}$.

$$\boldsymbol{\Omega = \{(2,3)\}}$$

Iteración 8…

Paso 2:

$s_{(2,3)} = 10$

$p_{(2,3)} = 3$

$f_{(2,3)} = 10 + 3 = 13$

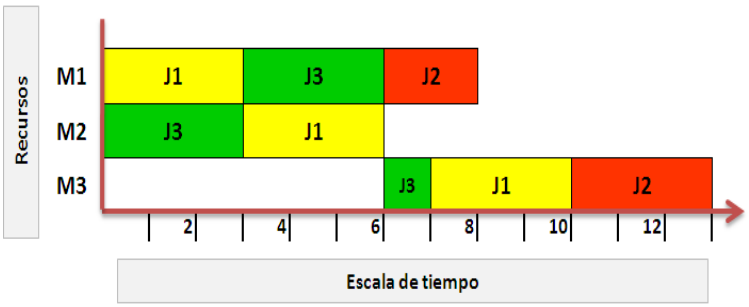
- $m^* = 3$

Paso 3:

1) $\{(2,3)\}$

2) $\{(2,3)\}$

3) Añadir $(2,3)$ a $S \rightarrow S = \{(1,1),(3,2),(3,1),(1,2),(3,3),(2,1),(1,3),(2,3)\}$



4) $s_{(2,3)} = f_{(2,3)} = 13$

Paso 4: Como no se ha generado un schedule completo, se borra $(2,3)$ de Ω. y se incluye su sucesor inmediato.

$$\boldsymbol{\Omega} = \{(\mathbf{2}, \mathbf{2})\}$$

Iteración 9…

Paso 2:

$s_{(2,2)} = 13$

$p_{(2,2)} = 4$

$f_{(2,2)} = 13 + 4 = 17$

- $m^* = 2$

Paso 3:

1) $\{(2,2)\}$

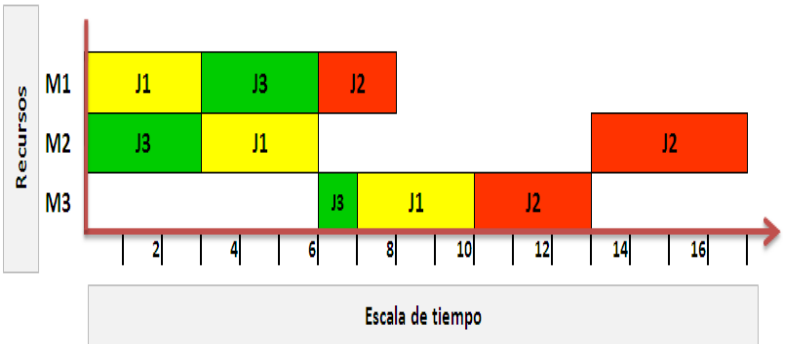
2) $\{(2,2)\}$

3) Añadir $(2,2)$ a $S \rightarrow S = \{(1,1), (3,2), (3,1), (1,2), (3,3), (2,1), (1,3), (2,3), (2,2)\}$

4) $s_{(2,2)} = f_{(2,2)} = 17$

Paso 4: Se ha generado un schedule completo, entonces el algoritmo se detiene.

$$\boldsymbol{\Omega} = \{\emptyset\}$$



Anexo B. Algoritmo de intercambio de Sha y Hsu

Se consideran las constantes $C1 = 0.6$ y $C2 = 0.2$, y las matrices x^1, v^1, $pbest_j^1$ y $gbest_j$ como sigue;

$$x^1 = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix} \qquad v^1 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

$$pbest_j^1 = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 3 & 1 \\ 2 & 3 & 1 \end{bmatrix} \qquad gbest_j = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 3 & 1 \\ 2 & 3 & 1 \end{bmatrix}$$

Máquina 1:

$\boldsymbol{x_{i1}^1} = [1 \quad 3 \quad 2]$ $\boldsymbol{v_{i1}^1} = [-1 \quad 1 \quad 0]$ $\boldsymbol{pbest_1^1} = [1 \quad 3 \quad 2]$ $\boldsymbol{gbest_1} = [1 \quad 3 \quad 2]$

Paso 1: Se escoge aleatoriamente una posición ζ de $x_{i1}^1 \rightarrow$

$\underline{\boldsymbol{\zeta = 2}}$

Paso 2: Se marca el trabajo en la posición 2 de x_{i1}^1 por $\Lambda_1 \rightarrow \Lambda_1 = 3$

Paso 3:

- Se genera un número aleatorio $rand = 0.451 < C1 = 0.6$, entonces se busca la posición de Λ_1 en $pbest_j^1$
- Se denota la posición que se ha encontrado en $pbest_j^1$ por $\zeta' \rightarrow \zeta' = 2$
- Se denota el trabajo en la posición $\zeta' = 2$ de x_{i1}^1 por $\Lambda_2 \rightarrow \Lambda_2 = 3$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 3$, no se realiza el intercambio.

Paso 5: Como no se han considerado todas las posiciones de x_{i1}^1, y $\zeta < n$ $(\zeta = 2 < n = 3)$ entonces $\zeta \leftarrow \zeta + 1$

$\zeta = 3$

Paso 2: $\Lambda_1 = 2$

Paso 3:

- $rand = 0.06 < C1$, entonces se busca Λ_1 en $pbest_j^1$
- $\zeta' = 3$
- $\Lambda_2 = 2$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 2$, no se realiza el intercambio.

Paso 5: Como no se han considerado todas las posiciones de x_{i1}^1, entonces $\zeta = 1$.

$\zeta = 1$

Paso 2: $\Lambda_1 = 1$

Paso 3:

- $rand = 0.226 < C1$, entonces se busca Λ_1 en $pbest_j^1$
- $\zeta' = 1$
- $\Lambda_2 = 1$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 1$, no se realiza el intercambio.

Paso 5: Como ya se han considerado todas las posiciones de x_{i1}^1, entonces el procedimiento de detiene y se continua con las demás máquinas.

Máquina 2:

$x_{i2}^1 = [2 \quad 1 \quad 3] \qquad v_{i2}^1 = [0 \quad 0 \quad 1] \qquad pbest_2^1 = [2 \quad 3 \quad 1] \qquad gbest = [2 \quad 3 \quad 1]$

$\boldsymbol{\zeta = 1}$

Paso 2: $\Lambda_1 = 2$

Paso 3:

- $rand = 0.137 < C1 \rightarrow$ se busca Λ_1 en $pbest_j^1$
- $\zeta' = 1$
- $\Lambda_2 = 2$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 2$, no se realiza el intercambio.

Paso 5: Como no se han considerado todas las posiciones de x_{i2}^1, y $\zeta < n$ $(\zeta = 1 < n = 3)$ entonces $\zeta \leftarrow \zeta + 1$

$\boldsymbol{\zeta = 2}$

Paso 2: $\Lambda_1 = 1$

Paso 3:

- $rand = 0.354 < C1$, entonces se busca Λ_1 en $pbest_j^1$
- $\zeta' = 3$
- $\Lambda_2 = 3$

Paso 4: $v_{i\Lambda_1}^1 = 0 \quad v_{i\Lambda_2}^1 = 1$, como $v_{i\Lambda_2}^1 \neq 0$ no se realiza el intercambio

Paso 5: Como no se han considerado todas las posiciones de x_{i2}^1, entonces $\zeta = \zeta + 1$.

$\underline{\boldsymbol{\zeta = 3}}$

Paso 2: $\Lambda_1 = 3$

Paso 3:

- $rand = 0.623 > C1$, entonces se busca Λ_1 en $gbest_j^1$
- $\zeta' = 2$
- $\Lambda_2 = 1$

Paso 4: $v_{i\Lambda_1}^1 = 1 \quad v_{i\Lambda_2}^1 = 0$, como $v_{i\Lambda_1}^1 \neq 0$ no se realiza el intercambio.

Paso 5: Como ya se han considerado todas las posiciones de x_{i2}^1, entonces el procedimiento se detiene y se continua con las demás máquinas.

Máquina 3:

$x_{i3}^1 = [2 \quad 3 \quad 1] \qquad v_{i3}^1 = [0 \quad -1 \quad 0] \qquad pbest_3^1 = [2 \quad 3 \quad 1] \qquad gbest = [2 \quad 3 \quad 1]$

$\underline{\zeta = 2}$

Paso 2: $\Lambda_1 = 3$

Paso 3:

- $rand = 0.185 < C1 \rightarrow$ se busca Λ_1 en $pbest_j^1$
- $\zeta' = 2$

- $\Lambda_2 = 3$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 3$, no se realiza el intercambio.

Paso 5: Como no se han considerado todas las posiciones de x_{i3}^1, y $\zeta < n$ $(\zeta = 2 < n = 3)$ entonces $\zeta \leftarrow \zeta + 1$

$\zeta = 3$

Paso 2: $\Lambda_1 = 1$

Paso 3:

- $rand = 0.671 > C1$, entonces se busca Λ_1 en $gbest$
- $\zeta' = 3$
- $\Lambda_2 = 1$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 1$, no se realiza el intercambio.

Paso 5: Como no se han considerado todas las posiciones de x_{i3}^1, entonces $\zeta = 1$.

$\zeta = 1$

Paso 2: $\Lambda_1 = 2$

Paso 3:

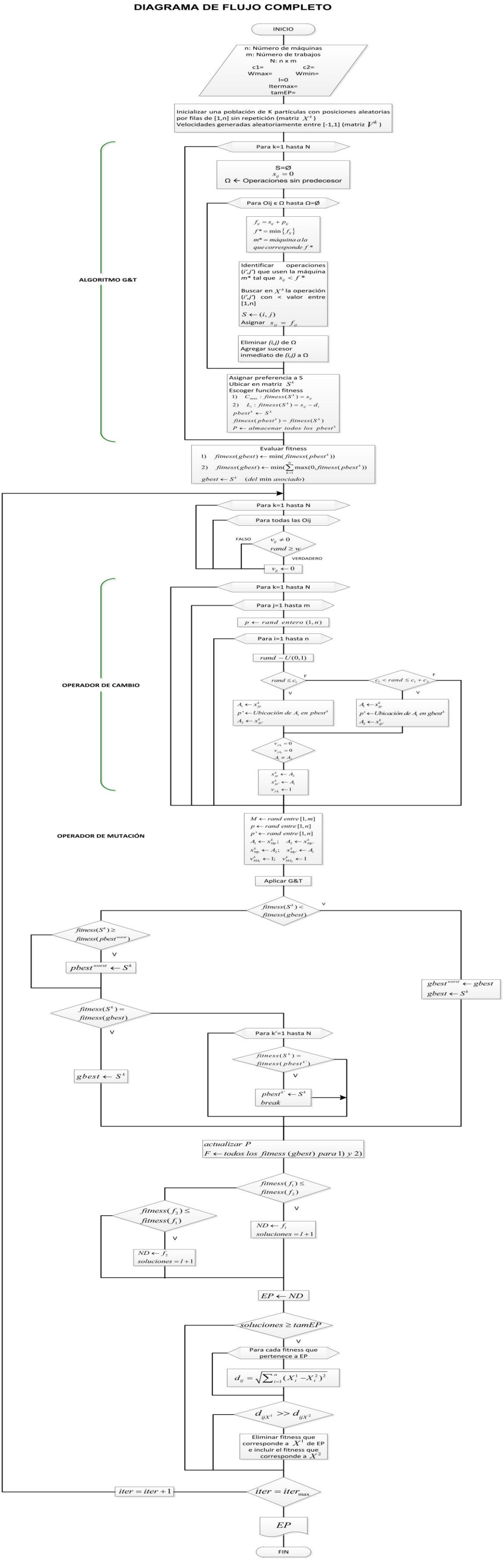
- $rand = 0.221 < C1$, entonces se busca Λ_1 en $pbest_j^1$
- $\zeta' = 1$
- $\Lambda_2 = 2$

Paso 4: Dado que $\Lambda_1 = \Lambda_2 = 2$, no se realiza el intercambio.

Paso 5: Como ya se han considerado todas las posiciones y todas las máquinas

$$x^1 = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix} \qquad v^1 = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$$

Anexo C. Diagrama de flujo MOPSO



Fuente: SARMIENTO, CINDY, 2012.