



Universidad Popular Autónoma del Estado de Puebla

Centro Interdisciplinario de Posgrados,

Investigación y Consultoría

Escuela de Ingeniería

Doctorado en Logística y Dirección de la Cadena de Suministro

Título de Tesis

**Diseño de una Cadena de Suministro con decisiones de Eficiencia,
Localización e Inventario usando un Algoritmo Evolutivo Multiobjetivo**

Tesis que para obtener el Grado de Doctor
en Logística y Dirección de la Cadena de Suministro

Presenta

Rodolfo Eleazar Pérez Loaiza

Puebla, México.

Enero 2015



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.



Universidad Popular Autónoma del Estado de Puebla

Centro Interdisciplinario de Posgrados,

Investigación y Consultoría

Escuela de Ingeniería

Doctorado en Logística y Dirección de la Cadena de Suministro

Se aprueba la Tesis:

**Diseño de una Cadena de Suministro con decisiones de Eficiencia,
Localización e Inventario usando un Algoritmo Evolutivo Multiobjetivo**

Comité de Revisión.

A handwritten signature in black ink, appearing to be 'E. Olivares', is written over a horizontal line.

Dr. Elías Olivares Benítez

Director

A handwritten signature in black ink, appearing to be 'J. L. Martínez', is written over a horizontal line.

Dr. Jose Luis Martínez Flores

Asesor

A handwritten signature in black ink, appearing to be 'P. Miranda', is written over a horizontal line.

Dr. Pablo Miranda González

Asesor

Puebla, México.

Enero 2015

AGRADECIMIENTOS

A DIOS:

POR LA OPORTUNIDAD DE VIVIR EN COMPAÑÍA DE MIS SERES QUERIDOS Y EL HABER CONCLUIDO ESTOS ESTUDIOS DOCTORALES SATISFACTORIAMENTE.

A MIS PADRES:

*EMMA LOAIZA SALDAÑA Y
ELEAZAR PEREZ CARRILLO*

QUIENES SIEMPRE ME HAN AYUDADO Y APOYADO A LO LARGO DE MI VIDA, EN LOS MOMENTOS ALEGRES Y DIFICILES. LES AGRADEZCO LA EDUCACION QUE FORJARON EN MI, PARA LLEGAR A SER UNA PERSONA DE BIEN.

A MI ESPOSA E HIJOS:

*MARTHA BELLO GARDUÑO,
RICARDO PEREZ BELLO Y
ROBERTO PEREZ BELLO*

MI ESPOSA QUIEN ES LA MUJER DE MI VIDA, QUE SIEMPRE HA ESTADO CONMIGO EN TODO MOMENTO, DANDOME SU APOYO, COMPRENSION, CARIÑO Y AMOR EN LAS BUENAS Y EN LAS MALAS, Y QUE JUNTO CON MIS HIJOS SON MI MOTIVACION EN LA VIDA, PARA SEGUIR ADELANTE.

A MI ASESOR:

DR. ELIAS OLIVARES BENITEZ

UNA GRAN PERSONA A QUIEN ESTIMO Y ADMIRO, PORQUE HA SIDO DE GRAN EJEMPLO DURANTE EL DESARROLLO DE ESTOS ESTUDIOS DOSTORALES, Y A QUIEN AGRADEZCO SU APOYO MORAL Y ENSEÑANZA PARA CONCLUIER ESTE PROYECTO.

CONTENIDO

	Pag.
CONTENIDO	1
LISTA DE FIGURAS	3
LISTA DE TABLAS	4
RESUMEN	5
 1. <u>INTRODUCCION</u>	6
1.1. Antecedentes	6
1.2. Justificación	7
1.3. Objetivos	9
1.4. Metodología	9
 2. <u>MARCO TEORICO</u>	10
2.1. Cadena de Suministro	10
2.2. Problemas de optimización Multiobjetivo	11
2.3. Selección de Proveedores en un Problema de Localización-Inventario para una cadena de Suministro (SS-ILPSC)	13
2.4. Metaheurísticas	17
2.5. Algoritmo Genético Multiobjetivo	18
2.6. Algoritmos Evolutivos en Diseño de Cadenas de Suministro	20
 3. <u>DISEÑO DE UNA CADENA DE SUMINISTRO CON DECISIONES DE EFICIENCIA, LOCALIZACIÓN E INVENTARIO USANDO UN ALGORITMO EVOLUTIVO MULTIOBJETIVO</u>	22
1. Introducción	22
2. Revisión de Literatura	23
3. Descripción del Problema y Modelo Matemático	24
3.1. Variables, parámetros y notación del modelo	24
4. Metodología	26
5. Experimentación y resultados	27
6. Conclusiones	28
7. Reconocimientos	28
8. Referencias	28

	Pag.
4. SUPPLY CHAIN DESIGN WITH EFFICIENCY, LOCATION AND INVENTORY POLICY USING A MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM	30
1. Introduction	30
2. Literature Review	31
3. Description of the Problem and Mathematical Model	32
3.1 Model variables, parameters and annotation	33
4. MOEA approach and epsilon-constraint method using LINGO	36
4.1 MOEA Method	36
4.1.1 Solution representation	36
4.1.2 Proposed Pareto Ranking	37
4.1.3 Preservation of Diversity	37
4.1.4 Selection, Crossover and Mutation Parameters	39
4.1.5 Coding	40
4.2 Epsilon-constraint method using LINGO	41
5. Experimentation and Results	42
5.1 Approximated solution sets using LINGO	42
5.2 Approximate solution sets using the Evolutionary Algorithm	42
5.3 Comparisons	43
6. Conclusions	46
7. Acknowledgements	47
References	47
5. CONCLUSIONES Y RECOMENDACIONES	50
BIBLIOGRAFIA	52
APENDICES	56
APENDICE A. Código en MATLAB del MOGA	56
APENDICE B. Código en LINGO ver.14.0	78
APENDICE C. Certificado CLAIO 2014	81
APENDICE D. Carta de Recepción del editor del artículo en Journal	82

LISTA DE FIGURAS

	Pag.
2. <u>MARCO TEORICO</u>	
Fig. 1. Una cadena de suministro	10
Fig. 2. Método ϵ -constraint	12
Fig. 3. Política de revisión Continua de Inventario	14
Fig. 4. MOGA dominación de aptitud	19
Fig. 5. MOGA Pseudocódigo	19
 3. <u>DISEÑO DE UNA CADENA DE SUMINISTRO CON DECISIONES DE EFICIENCIA, LOCALIZACIÓN E INVENTARIO USANDO UN ALGORITMO EVOLUTIVO MULTIOBJETIVO</u>	
Fig. 1. Esquema de Distribución del modelo SS-ILPSC.....	24
Fig. 2. Frente de Pareto usando distancia euclidiana	27
 4. <u>SUPPLY CHAIN DESIGN WITH EFFICIENCY, LOCATION AND INVENTORY POLICY USING A MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM</u>	
Fig. 1. Distribution of the SS-ILPSC model.	33
Fig. 2. Solution representation	37
Fig. 3. Crossover	39
Fig. 4. Mutation	40
Fig. 5. Multi-objective Evolutionary Algorithm	40
Fig. 6. Procedure of the Multi-objective Evolutionary Algorithm	41
Fig. 7. Metric of Euclidian distance between a reference point and points on the front.	43
Fig. 8. Comparison of fronts for group scenario 5-3-10-15	46

LISTA DE TABLAS

	Pag.
3. DISEÑO DE UNA CADENA DE SUMINISTRO CON DECISIONES DE EFICIENCIA, LOCALIZACIÓN E INVENTARIO USANDO UN ALGORITMO EVOLUTIVO MULTIOBJETIVO	
Tabla 1. Comparación de réplicas en términos de su media y desviación estándar. 	28
4. SUPPLY CHAIN DESIGN WITH EFFICIENCY, LOCATION AND INVENTORY POLICY USING A MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM	
Tabla. 1. Comparison of results with LINGO and the MOEA for scenario 5-3-5-10. 	44
Tabla. 2. Comparison of results with LINGO and the MOEA for scenario 5-3-10-10. 	44
Tabla. 3. Comparison of results with LINGO and the MOEA for scenario 5-3-10-15. 	45
Tabla. 4. Comparison of results with LINGO and the MOEA for scenario 5-5-5-15. 	45
Tabla. 5. Comparison of results with LINGO and the MOEA for scenario 5-3-10-15. 	45

RESUMEN

En este trabajo se presenta el desarrollo de una metaheurística basada en un algoritmo evolutivo multiobjetivo para resolver el modelo bi-objetivo de programación no lineal entero-mixto para el diseño de una cadena de suministro con decisiones de localización-inventario y selección de proveedores. En el modelo los proveedores envían las materias primas a un conjunto de plantas manufactureras para su procesamiento, éstas envían el producto terminado a un conjunto de centros de distribución, para que posteriormente sean reenviados a un conjunto de minoristas. Las variables de decisión son la apertura de plantas y centros de distribución, considerando una política de inventario de revisión continua, y el flujo de materiales entre las diferentes instalaciones. Los objetivos en conflicto son: minimizar los costos totales en los que se incurre en toda la cadena y la selección de los proveedores con el valor más alto combinado respecto a una métrica denominada OEE (*Overall Equipment Effectiveness*). Se resuelven instancias de pequeño y mediano tamaño. Los resultados son comparados con los frentes de Pareto óptimos obtenidos con un software comercial de optimización usando el método de epsilon constraint. Los resultados numéricos demuestran la efectividad computacional de la metaheurística propuesta.

1. INTRODUCCION

1.1 Antecedentes

Una cadena de suministro esta conformada por varios integrantes, que están involucrados directa o indirectamente, para satisfacer la demanda de un cliente. Estos son los fabricantes, proveedores, transportistas, almacenes y minoristas principalmente.

La clave del éxito de las compañías de hoy en día radica en la integración de las actividades, la cooperación, la coordinación y el intercambio de información desde los proveedores hasta los clientes, a través de una gestión eficaz, con el propósito de reducir costos y mejorar el servicio.

En las últimas décadas la logística y la cadena de suministro en las organizaciones, han sido temas centrales de investigación en la industria y en la academia. El objetivo principal es responder y dar solución a los problemas que se presentan, ocasionados por un mercado competitivo y global. Se requiere el uso de sistemas adecuados de soporte de decisiones para cada organización, basados en modelos matemáticos y técnicas de solución eficaces, en conjunto con el uso de tecnologías de la información y comunicaciones.

Con el uso de metaheurísticas se han logrado obtener soluciones rápidas a problemas complejos en las cadenas de suministro de las organizaciones, en el diseño y gestión de toda la cadena como una sola entidad.

En este caso se incluirá el problema de selección de proveedores en un problema de localización- inventario para una cadena de suministro de tres niveles a partir del modelo propuesto por (Guerrero, 2013). Este problema consiste en el diseño de una cadena de suministro conformada por tres escalones, en donde el primer escalón, los proveedores, envían las materias primas a un conjunto de plantas manufactureras; en el segundo escalón, las plantas manufactureras envían el producto terminado a un conjunto de almacenes; y en el tercer escalón se envía el producto de los almacenes a un conjunto de clientes finales o minoristas.

El número de proveedores, plantas, almacenes y clientes es conocido, las localizaciones y número de plantas y almacenes no están definidos, y la asignación de los minoristas se basa en la minimización de los costos y la maximización de la efectividad global del equipo. Cada posible planta y almacén tienen asignado un costo fijo y una capacidad limitada.

El modelo incorpora las operaciones tácticas junto con las decisiones estratégicas, para la selección de proveedores (basado en la efectividad global del equipo y los costos) y la localización-inventario, así como la determinación del flujo entre las diferentes instalaciones en cada uno de los escalones de la red, considerando la cantidad de productos en inventario con respecto a la demanda y a su varianza. Se asume una política de revisión continua de inventarios, bajo un ambiente de integración de riesgo.

Para resolver lo anterior se considera una formulación matemática bi-objetivo, donde uno de los objetivos es la maximización del OEE (Overall Equipment Effectiveness) de acuerdo a la selección de proveedores y la minimización del costo total de la cadena de suministro considerando un nivel de servicio.

Este problema es resuelto con la técnica ϵ -constraint para instancias pequeñas usando el software de optimización comercial LINGO ver 14.0. Debido a la naturaleza no lineal del modelo, las soluciones obtenidas son óptimos locales, el solucionador que utiliza el software comercial está basado en el método de gradiente reducido generalizado con programas lineales sucesivos. Lo que se pretende con esta investigación es presentar una metaheurística para resolver este modelo de programación matemática no lineal, basado en un algoritmo evolutivo multiobjetivo como método de optimización aproximado, para encontrar los frentes de Pareto en instancias pequeñas y grandes.

1.2 Justificación

Las organizaciones de hoy en día tienen un comportamiento más dinámico, buscando adecuarse a los mercados globalizados, que constantemente están cambiando, bajo un ambiente de incertidumbre. Los gerentes deben constantemente estar mirando su cadena de suministro para ofrecer soluciones que mejoren los tiempos de entrega y disminuyan los costos, donde se ofrecen una alta variedad de productos con un ciclo de vida más corto.

Se puede mencionar entonces que es de suma importancia el método que se utilice para generar soluciones óptimas y factibles al tipo de problemas que se presentan en las organizaciones, y más aún, cuando las variables que se tienen que controlar son bastantes o que impactan de manera considerable en el desempeño de toda la cadena.

El uso de técnicas metaheurísticas en la actualidad es capaz de proporcionar rápidamente soluciones casi óptimas a problemas donde la optimización exacta está limitada, en una cantidad razonable de tiempo.

Teniendo en cuenta que los problemas de la cadena de suministro generalmente exhiben complejidad significativa en el mundo real, las metaheurísticas son especialmente adecuadas para la evaluación de estas dificultades. Además de manejar problemas de gran tamaño, las metaheurísticas también puede generar soluciones a problemas complejos cuando las restricciones o la función objetivo no son continuas o diferenciables, por ejemplo, cuando los precios de transporte para los envíos cambian con la distancia y la cantidad enviada (Galbreth et al. 2008).

Las metaheurísticas también proveen un método robusto que puede ser adoptado a problemas con diferentes características de solución. Esta capacidad es importante cuando nos encontramos con problemas de la cadena de suministro, restricciones y condiciones que pueden cambiar frecuentemente. El diseño de los algoritmos de las metaheurísticas, también los hacen actualizarse cuando se introducen al mundo real y ocurren cambios (Stanley E. et al. 2012).

Además, como las metaheurísticas realizan una búsqueda amplia de las soluciones posibles para un problema en específico, tienen la capacidad de encontrar soluciones óptimas alternativas y proporcionar al usuario opciones diferentes. Esto es especialmente útil en los problemas de logística que pueden tener varias soluciones alternativas, cada una con un resultado óptimo equivalente (costo, distancia, velocidad, etc.)

De esta forma, las metaheurísticas pueden proporcionar una variedad de soluciones a los problemas multiobjetivo, permitiendo que los resultados de estas soluciones se utilicen para desarrollar un frente de Pareto (McMullen & Tarasewich, 2006), por lo que el usuario puede identificar las ventajas y desventajas en objetivos tales como servicio al cliente o el costo.

El modelo de selección de proveedores en un problema de localización- inventario para una cadena de suministro de tres niveles (Guerrero, 2013) será resuelto mediante una metaheurística basada en la utilización de un algoritmo genético multiobjetivo (MOGA).

Una revisión inicial de este modelo se ha resuelto mediante el método ϵ -constraint a través de una linealización, y se ha encontrado que para instancias grandes éste no converge en un tiempo razonable, además de que en algunos casos no se obtienen soluciones factibles (Guerrero, 2013).

1.3 Objetivos

Objetivo General:

El objetivo de esta investigación es resolver el modelo de selección de proveedores en un problema de localización-inventario para una cadena de suministro de tres niveles (Guerrero, 2013), mediante la aplicación de una metaheurística basada en un algoritmo genético multiobjetivo considerando la variabilidad de la demanda de los clientes finales, buscando minimizar los costos de transportación y los costos fijos de las plantas y almacenes, y también maximizar la eficiencia en la selección de los proveedores medida con el indicador OEE(*Overall Equipment Effectiveness*) .

Objetivos específicos:

- Revisión de literatura de trabajos donde se aborden los problemas de diseño de cadenas de suministro considerando la selección de proveedores y decisiones de control de inventario y sus métodos de solución.
- Desarrollar una metaheurística en base a un algoritmo genético multiobjetivo (MOGA) para resolver este problema.
- Comparar los resultados obtenidos con un software de optimización comercial para instancias pequeñas.
- Generar y resolver diferentes instancias grandes.
- Análisis y comparar los resultados obtenidos.

1.4 Metodología

1. Análisis del problema propuesto.
2. Realizar un análisis bibliográfico sobre este tema.
3. Identificar la información relevante para la investigación.
4. Conceptualizar la técnica MOGA de Computación Evolutiva.
5. Desarrollar el pseudocódigo de la técnica MOGA para solucionar el problema propuesto.
6. Diseñar las diferentes instancias (grandes y pequeñas) para validar la técnica MOGA.
7. Probar la técnica MOGA con las diferentes instancias.
8. Análisis de resultados y conclusiones.

2. MARCO TEORICO

2.1 Cadena de Suministro

Una cadena de suministro es un sistema logístico complejo en el cual la materia prima es transformada en productos terminados y después distribuidos a los usuarios finales (consumidores o compañías). Esto incluye proveedores, centros de manufactura, almacenes, centros de distribución y puntos de venta. En la figura 1, se muestra una cadena de suministro tradicional en donde la producción y los sistemas de distribución están compuestos de dos etapas. (Ghiani et al. 2004).

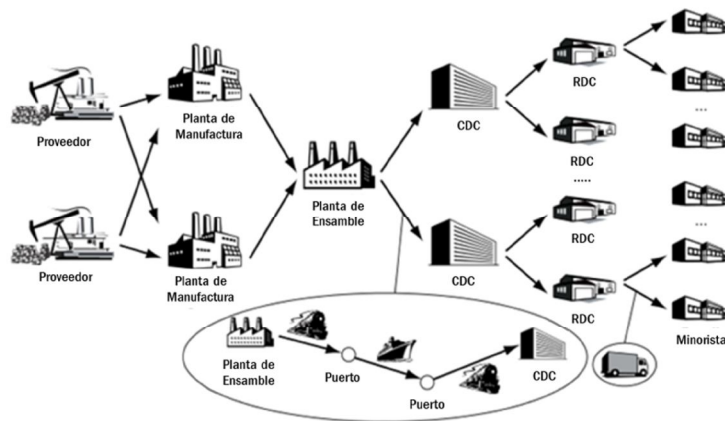


Fig. 1. Una cadena de suministro. Fuente: Ghiani et al. (2004)

El diseño de la cadena de suministro es uno de los más grandes problemas de decisión estratégica, por lo que son usadas para las eficientes operaciones a largo plazo en toda la cadena, y por esta razón es necesario optimizar (Taskin et al. 2009). Consiste en determinar como elegir las entidades y distribución de bienes, para satisfacer las demandas de los clientes con un costo total mínimo (Ji & Zhao, 2006), lo que significa encontrar las respuestas para las decisiones en el panorama a largo plazo, tales como: ¿Dónde podrían las plantas o almacenes ser localizados?, ¿Podrían las instalaciones existentes ser utilizadas o construir otras nuevas?, ¿Qué productos podrían ser producidos y en que plantas? ¿Qué cantidad de los productos deben ser producido y por qué? ¿Cuánto inventario se debe mantener en stock? ¿Cuanta materia prima, productos semiterminados y productos terminados? ¿Cuándo es mejor mover las mercancías? ¿Qué modo de transporte se debe usar? (Li & Schulze, 2011).

2.2 Problemas de Optimización Multiobjetivo

Cuando se presenta la necesidad de resolver problemas de optimización, las buenas decisiones de basan en un solo criterio. Sin embargo, gran parte de los problemas del mundo real implican la optimización simultánea de varios objetivos que generalmente presentan conflictos entre ellos; es decir, la mejora en uno conduce a un deterioro en el otro. Por ejemplo escoger el medio ideal para llegar al trabajo, establecer el orden de nuestras tareas, elegir el restaurante para el almuerzo, hacer las compras en el supermercado, preparar la cena y la distribución de actividades en el tiempo de ocio restante.

Aunque la mayoría de los problemas de decisión involucran este tipo de situaciones, las propuestas computacionales de automatización que se han presentado para resolverlos habitualmente se limitan a convertir el problema de objetivos múltiples en uno en que existe un solo objetivo. Esta reducción es por los modelos matemáticos empleados y puede realizarse de varias maneras, por ejemplo se prioriza uno de los objetivos y los demás se colocan como restricciones, o también se genera un objetivo compuesto otorgando pesos a los objetivos en juego y armando una suma ponderada de los mismos. De todos modos, ninguna de estas reducciones refleja fielmente al problema y, por tanto, tampoco otorga soluciones completamente satisfactorias.

El enfoque de optimización multiobjetivo conocido como Multiobjective Optimization Problem (MOP) se puede definir, en palabras, como el problema de hallar una solución que satisfaciendo ciertas restricciones, optimiza varias funciones objetivos simultáneamente. Estas funciones usualmente en conflicto conforman una descripción del criterio de desempeño. Por lo tanto no hay una sola solución sino un conjunto de soluciones que son encontradas a través del uso de la teoría de optimalidad de Pareto (Coello et al. 2007).

La forma general de un problema de optimización multiobjetivo es la siguiente (Deb, 2008):

$$\begin{array}{lll}
 \text{Minimizar/Maximizar} & f_m(x), & m = 1, 2, \dots, M; \\
 \text{Sujeto a} & g_j(x) \geq 0, & j = 1, 2, \dots, J; \\
 & h_k(x) = 0, & k = 1, 2, \dots, K; \\
 & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \dots, n;
 \end{array}$$

Una solución x es un vector de n variables de decisión $x = (x_1, x_2, \dots, x_n)^T$. El último conjunto de restricciones son llamadas límite de variables, restringiendo cada variable x_i a tomar valores dentro de los límites de $x_i^{(L)}$ y $x_i^{(U)}$. Estos límites constituyen un espacio de variables de decisión D o espacio de decisión. Los términos $g_j(x)$ y $h_k(x)$ son llamadas funciones de

restricción, que pueden ser del tipo “mayor o igual que”, “menor o igual que” o “igual que”.

Uno de los métodos clásicos para resolver problemas de optimización multiobjetivo, es el de épsilon-constraint, el cual genera soluciones no dominadas y la persona que toma decisiones, elige una de ellas.

El método épsilon-constraint consiste en la creación de un modelo con un solo objetivo que es optimizado y las demás funciones se convierten en restricciones en el modelo (Ehrgott, 2005). Tal como se muestra a continuación, un modelo de optimización multiobjetivo puede ser reformulado a través de un modelo épsilon-constraint como sigue:

$$\begin{array}{ll}
 \text{Minimizar/Maximizar} & f_i(X) \\
 \text{Sujeto a} & f_k(X) \leq \epsilon_k, \quad k = 1, \dots, n \text{ y } k \neq i \\
 & H(x) = 0 \\
 & G(X) \leq 0
 \end{array}$$

En este caso, la función $f_i(X)$ es la única en ser optimizada; las otras $(n - 1)$ funciones se convierten en restricciones y son limitadas por su valores correspondientes. El objetivo en este método consiste en cambiar los valores ϵ en cada una de las funciones y, de esta forma obtener diferentes valores optimizados en la función $f_i(X)$.

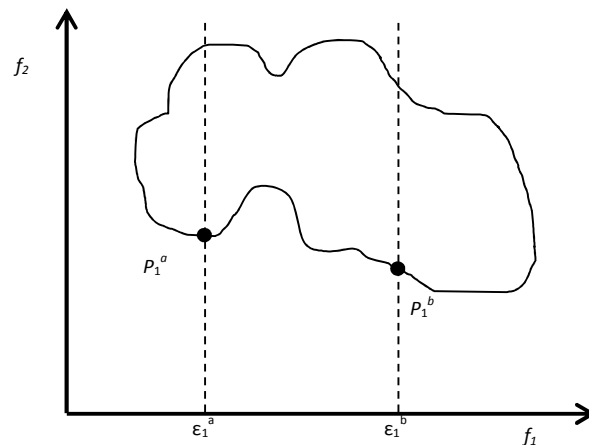


Fig. 2. Método épsilon-constraint

En la figura 2 se muestra un ejemplo en el cual dos funciones (f_1 y f_2) son optimizadas utilizando el método de épsilon-constraint. Aquí es optimizada y se convierte en restricción. Para las dos soluciones P_1^a y P_1^b se puede observar en la figura que la función f_1 es una restricción con un límite superior ϵ_1^a ; el punto de optimización

encontrado para f_2 es P_1^a . Cuando en el modelo para el valor de la restricción f_1 cambia a ε_1^b , el valor optimizado encontrado para f_2 es P_1^b . Si cambiamos el valor de la restricción para f_1 , obtenemos diferentes valores para f_2 , y de esta forma obtenemos los valores para ε_1^b , el frente de Pareto optimo.

2.3 Selección de Proveedores en un Problema de Localización-Inventario para una cadena de Suministro (SS-ILPSC)

Las decisiones de diseño de redes determinan la configuración de la cadena de suministro y tienen un impacto significativo en los costos logísticos y capacidad de respuesta (Chopra & Meindl, 2008), por lo que una oportunidad es investigar la influencia que tienen los canales de transportación.

El modelo de selección de proveedores en un problema de localización-inventario para una cadena de suministro (Guerrero, 2013), es un planteamiento de diseño no lineal entera mixta bi-objetivo, que puede ser considerado una extensión del problema de localización de instalación capacitado CFLP, clasificado como NP-duro. La política de inventario y la asignación en una cadena de suministro de tres niveles, introducen la no linealidad en el modelo. El diseño de la cadena de suministro se compone de proveedores, plantas, almacenes, y los minoristas, teniendo en cuenta un solo producto y todos los minoristas tienen una demanda y la varianza independiente.

Suponemos que la ubicación de los proveedores, plantas y almacenes son conocidos y tienen una capacidad. Los proveedores tienen un indicador de rendimiento - Efectividad global del equipo (OEE) que es una métrica que supervisa el rendimiento real de los equipos en relación a su capacidad de funcionamiento en condiciones óptimas de operación, el cual considera las seis grandes pérdidas en el proceso de fabricación: pérdida de avería, pérdida de instalación y ajuste, pérdida de la velocidad reducida, pérdida de paros menores, pérdida de calidad y pérdida de puesta en marcha. La OEE integra estas seis grandes pérdidas en los tres parámetros fundamentales de la producción industrial: la disponibilidad, eficiencia y calidad. La disponibilidad tiene en cuenta la pérdida de tiempo en la producción prevista, por ejemplo, ajustes, la escasez de materia prima, tiempo de preparación, y fallas en los equipos, en muchos casos, esta pérdida de tiempo no puede ser eliminada, pero puede ser reducida. El rendimiento considera las variaciones de velocidad de fabricación, por ejemplo, paros menores, materiales de baja calidad y el funcionamiento ineficiente. La calidad tiene en cuenta la cantidad de productos elaborados que no cumple con los estándares de calidad, retrabajo y desperdicios.

$$OEE_i = \left(\frac{et_i}{st_i} \right) \cdot \left(\frac{ct_i \cdot tp_i}{et_i} \right) \cdot \left(\frac{tp_i - td_i}{tp_i} \right) \quad (1)$$

Donde:

- et_i : Tiempo de producción efectivo del proveedor i
- st_i : Tiempo de la producción programada del proveedor i
- ct_i : Tiempo de ciclo ideal del proveedor i
- tp_i : Total de productos producidos por el proveedor i
- td_i : Total de productos defectuosos del proveedor i

Los costos a considerar son: el costo de instalar plantas y almacenes, el costo de inventario en las instalaciones, que se relacionan con la política de inventarios y los costos de transporte entre los diferentes niveles de la cadena de suministro. que está en función de la asignación de los minoristas a cada almacén y de los almacenes a cada planta. Cada minorista debe ser asignado a un almacén.

La problemática es determinar el número óptimo de plantas y almacenes y su ubicación, así como la asignación de los almacenes a las plantas, y de los minoristas a los almacenes, además el flujo de producción de los proveedores a las plantas, buscando minimizar los costos totales en el sistema y maximizar la eficacia global del equipo (OEE) de los proveedores.

La política de revisión continua de inventario (q , RP) se utiliza en cualquier almacén y planta para satisfacer la demanda de un modelo estocástico, como generalmente se considera en los modelos de localización-inventario. La demanda de almacén depende de los clientes asignados a cada almacén k , y la demanda de la planta depende de los almacenes asignados a cada planta j . La figura 3 muestra el rendimiento del inventario. Cuando el nivel de inventario llega al punto de reorden (RP), una orden de q unidades de producto es generada y se reciben en un determinado tiempo (LT), con una probabilidad determinada. Esta política no penaliza a las demandas insatisfechas.

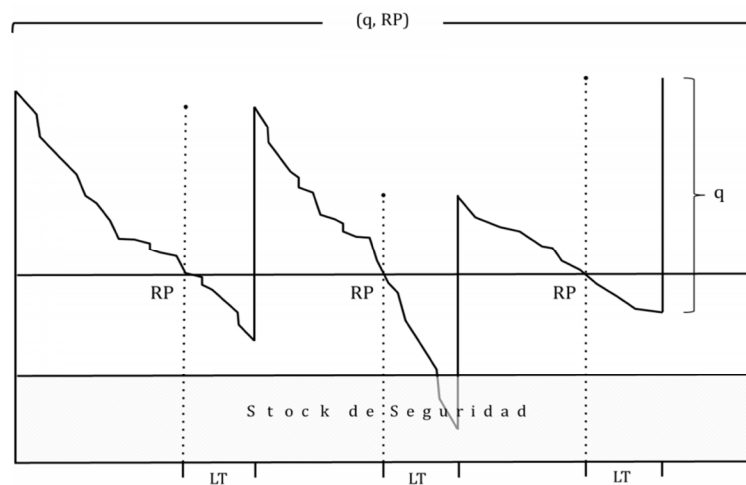


Fig. 3. Política de revisión Continua de Inventario. Fuente: Guerreo (2013)

Para calcular el costo de inventario de trabajo WIC (4) es considerado el costo total de inventario.

$$\text{Costo total de inventario} = cd + \left(\frac{d}{q}\right)oc - \left(\frac{q}{2}\right)hc \quad (2)$$

Donde cd es el costo del material, d es la demanda, q es la cantidad a ordenar, oc es el costo de ordenar, y hc es el costo de mantener inventario. Derivando (2) con respecto a q se obtiene la cantidad de orden económica q^* (3). Sustituyendo (3) en (2) se obtiene WIC (4), el cual es considerado en cada planta y almacén.

$$q^* = \sqrt{\frac{2 \cdot d \cdot oc}{hc}} \quad (3)$$

$$WIC = \sqrt{2 \cdot oc \cdot hc} \cdot \sqrt{d} \quad (4)$$

Asumiendo una distribución normal en la demanda, RP puede ser calculado de acuerdo a (5). Donde ZT representa el valor de la distribución Normal entre (0,1) que acumula una probabilidad de $1 - \alpha$. Esta probabilidad es conocida como nivel de seguridad (6) asociado con el stock de seguridad.

$$\text{Prob}(d(LT) \leq RP) \geq 1 - \alpha \quad (5)$$

$$RP = d \cdot LT \cdot ZT \cdot \sqrt{u} \cdot \sqrt{LT} \quad (6)$$

RP es la cantidad de inventario que podría estar disponible en cada planta y almacén para absorber fluctuaciones en la demanda del minorista durante el tiempo de entrega, y es compuesto por el promedio de la demanda en el tiempo de entrega $d \cdot LT$ y el stock de seguridad $ZT \cdot HC \cdot \sqrt{u} \cdot \sqrt{LT}$. El stock de seguridad presenta un balance entre el costo de no tener existencias y el costo de mantener stock de seguridad. En este caso no fue considerado el costo de no tener existencias solamente el costo del stock de seguridad (7) en cada planta y almacén.

$$SSC = ZT \cdot HC \cdot \sqrt{u} \cdot \sqrt{LT} \quad (7)$$

Se consideran dos funciones objetivos que son optimizadas simultáneamente, donde $Z1$ minimiza el costo total de trabajo WIC, el costo del stock de seguridad SSC, los costos fijos en los almacenes k y plantas j , los costes de transporte entre las etapas y el costo del volumen de producción β_{ij} . $Z2$ maximiza la efectividad global del equipo (OEE) de los proveedores al resto de las plantas.

$$\text{Donde: } \tau_j = \sqrt{2 \cdot OC_j \cdot HC_j}, \quad \phi_{ij} = Z_{1-\alpha} \cdot HC_j \cdot \sqrt{LT_{ij}}, \quad \psi_k = \sqrt{2 \cdot OC_k \cdot HC_k}, \quad \text{y}$$

$$\omega_{jk} = Z_{1-\alpha} \cdot HC_k \cdot \sqrt{LT_{jk}}.$$

Por lo tanto, el problema es formulado como sigue:

$$\begin{aligned} \text{Min } Z1 = & \sum_{j=1}^J \left(FCp_j \cdot y_j + \tau_{ij} \cdot \sqrt{\delta_j} + \sum_{i=1}^I \left(\varphi_{ij} \cdot \sqrt{\eta_j} + UC_i \cdot \beta_{ij} \right) \right) + \\ & \sum_{k=1}^K \left(FCw_k \cdot x_k + \psi_{jk} \cdot \sqrt{\lambda_k} + \sum_{j=1}^J \left(\omega_{jk} \cdot \sqrt{\pi_k} + A_{jk} \cdot \lambda_k \cdot z_{jk} \right) \right) + \sum_{k=1}^K \sum_{l=1}^L B_{kl} \cdot d_l \cdot w_{kl} \end{aligned} \quad (8)$$

$$\text{Max } Z2 = \sum_{i=1}^I \sum_{j=1}^J \left(\beta_{ij} \cdot OEE_i \right) \quad (9)$$

Subject to:

$$\sum_{k=1}^K w_{kl} = 1 \quad \forall l = 1, \dots, L \quad (10)$$

$$\sum_{j=1}^J z_{jk} = x_k \quad \forall k = 1, \dots, K \quad (11)$$

$$\sum_{l=1}^L d_l \cdot w_{kl} = \lambda_k \quad \forall k = 1, \dots, K \quad (12)$$

$$\sum_{l=1}^L u_l \cdot w_{kl} = \pi_k \quad \forall k = 1, \dots, K \quad (13)$$

$$\sum_{k=1}^K \delta_k \cdot z_{jk} = \delta_j \quad \forall j = 1, \dots, J \quad (14)$$

$$\sum_{k=1}^K \eta_k \cdot z_{jk} = \eta_j \quad \forall j = 1, \dots, J \quad (15)$$

$$\sum_{i=1}^I \beta_{ij} = \delta_j \quad \forall j = 1, \dots, J \quad (16)$$

$$\lambda_k \leq cw_k \cdot x_k \quad \forall k = 1, \dots, K \quad (17)$$

$$\delta_j \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (18)$$

$$\sum_{i=1}^I \beta_{ij} \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (19)$$

$$\sum_{j=1}^J \beta_{ij} \leq cs_i \quad \forall i = 1, \dots, I \quad (20)$$

$$w_{kl}, x_k, y_j, z_{jk} \in \{0, 1\} \quad \forall j = 1, \dots, J, \forall k = 1, \dots, K, \forall l = 1, \dots, L \quad (21)$$

$$\beta_{ij} \in \mathbb{R} \quad \forall i = 1, \dots, I, \forall j = 1, \dots, J \quad (22)$$

La ecuación (8) representa el costo total del sistema. La primera expresión representa el costo fijo si se abre la planta j . La segunda expresión representa el stock de seguridad de almacenamiento (SSC), el costo de inventario de trabajo (WIC) en cada planta, y el costo de las unidades producidas por el proveedor i y enviado a la planta j . La tercera expresión corresponde al costo fijo si se abre el almacén k . La cuarta expresión corresponde al stock de seguridad de almacenamiento (SSC), el costo de inventario de trabajo (WIC) en cada almacén y el costo de transporte de la planta j al almacén k . La última expresión es el costo de transporte del almacén k al minorista l . La ecuación (9) representa el nivel medio de la efectividad global del equipo (OEE) de las unidades producidas por el proveedor i y que son enviadas a la planta j .

Las restricciones (10) y (11) son de asignación. La restricción (10) asegura la asignación minorista a un único almacén y la restricción (11) asegura la asignación de un almacén a una planta. Las restricciones (12) y (13) agregan la demanda promedio y varianza de los minoristas a cada almacén.

De la misma manera, las restricciones (14) y (15) agregan la demanda y varianza de los almacenes a cada planta respectivamente. La restricción (16) establece que el total de producto producido por uno o más proveedores y enviado a cada planta es igual a la demanda en cada planta. Las restricciones de capacidad en almacenes, plantas, y los proveedores se presentan de la (17) a la (20).

La restricción (17) y (18) asegura que la demanda media de cada almacén o planta es menor o igual a su capacidad. La restricción (19) que establece total de producto producido por uno o más proveedores y enviado a cada planta es menor o igual a la capacidad de la planta. La restricción (20) asegura que el total de productos producidos para los proveedores i y enviado a una o más plantas, es menor o igual a la capacidad de producción del proveedor i . Finalmente las restricciones (21) y (22) muestran la naturaleza de las variables.

2.4 Metaheurísticas

Uno de los principales métodos analíticos utilizados para resolver una variedad de problemas, son los de optimización, los cuales son guiados por conjuntos de variables de decisión que se manipulan para alcanzar una solución óptima dentro de los confines de un conjunto de restricciones que prohíben que las variables asuman valores no factibles (Hillier & Hillier, 2009; Ragsdale, 2011). Sin embargo debido al tamaño, complejidad y con frecuencia la naturaleza no lineal de algunos problemas, los enfoques de forma cerrada que producen los métodos exactos para la solución de un problema no siempre están disponibles. El ampliamente empleado método simplex, por ejemplo, está limitado en el tamaño y estructura de los problemas que pueda resolver.

Por lo tanto, los métodos heurísticos han sido tradicionalmente utilizados para generar soluciones a los problemas de optimización grandes y complejos. Esta capacidad es importante cuando se enfrentan a problemas de la cadena de suministro, las limitaciones, y las condiciones que pueden cambiar con frecuencia. El diseño modular de los algoritmos de metaheurísticas también hace que sean fáciles de actualizar y volver a ejecutar cuando se producen los cambios en el mundo real (Lourenco, 2005).

Un ejemplo es el que presenta Olivares-Benitez et al. (2013), donde en su publicación presentan el algoritmo de una metaheurística que combina principios de funciones voraces (greedy), búsqueda dispersa (scatter search), reencadenamiento de trayectorias (parth relinking) y programación matemática, para resolver el problema de selección de canales de transportación en el diseño de una cadena de suministro que consta de dos escalones y un solo producto.

2.5 Algoritmo Genético Multiobjetivo (MOGA)

Carlos M. Fonseca y Peter J. Fleming introducen un algoritmo genético multiobjetivo (MOGA) el cual usa la clasificación no dominada de la población de un algoritmo genético. Ellos hacen mucho énfasis en las soluciones no dominadas y simultáneamente mantienen la diversidad en las soluciones no dominadas (Deb, 2008).

A cada individuo de la población se le asignará un rango de acuerdo al cual será ordenado para la selección. El rango se asigna según un criterio de no dominancia. Los individuos no dominados tendrán un rango de 1. Los individuos dominados son penalizados de acuerdo a la densidad de las regiones que lo dominan. Si x_i es no dominado entonces $\text{rango}(x_i) = 1$, en otro caso $\text{rango}(x_i) = 1 + (\text{número de individuos que lo dominan})$ (Fonseca & Fleming, 1993), es decir, el rango de un cierto individuo corresponde al número de individuos en la actual población por los cuales está dominado. En la figura 4 se muestran la variedad de puntos dominados.

El MOGA difiere del algoritmo genético estándar en la manera de calcular la aptitud de cada individuo de la población, utiliza el *shared fitness*, el resto del algoritmo es el mismo, utilizando el método de selección universal estocástica, un único punto de cruce y un bit de mutación. El pseudocódigo de MOGA, se representa en la figura 5.

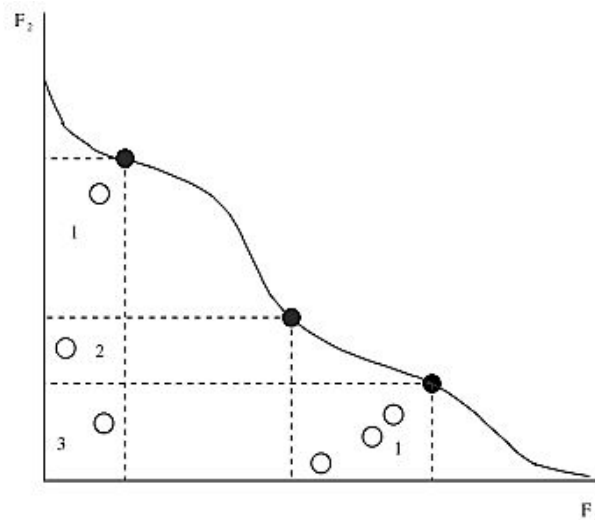


Fig. 4. MOGA dominación de aptitud. Fuente: Coello et al. (2007)

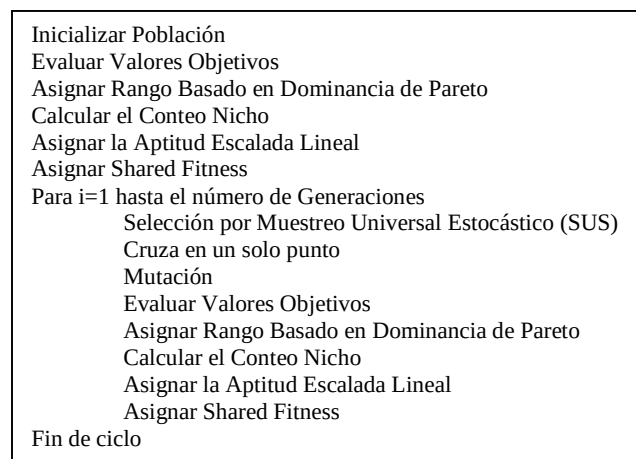


Fig. 5. MOGA Pseudocódigo. Fuente: Coello et al. (2007)

El procedimiento de asignación de aptitud es desarrollado como sigue (Fonseca & Fleming, 1993):

1. La población se ordena de menor a mayor de acuerdo al rango que se le ha asignado a cada individuo.
2. Se asigna el valor de fitness para cada individuo por interpolación desde el mejor (rango 1) hasta el peor.
3. Se promedia la adaptación de los individuos con el mismo rango, para que tengan el mismo valor de fitness.

2.6 Algoritmos Evolutivos en Diseño de Cadenas de Suministro

En la literatura de diseño de cadenas de suministro podemos encontrar varios trabajos que han empleado métodos heurísticos para buscar soluciones a los problemas planteados, específicamente los de computación evolutiva, debido a los logros significativos que se han obtenido en los últimos años.

Radhakrishnan & Natarajan (2011), en su publicación describen que la eficiente y efectiva administración de inventarios a través de la cadena de suministro, significativamente mejora el último servicio ofrecido al cliente. La naturaleza dinámica del exceso en el nivel de stock y en el nivel de escasez en todos los periodos, es un serio tema cuando la implementación es considerada. Proponen un nuevo enfoque de solución basado en un algoritmo genético en el cual el más probable exceso en el nivel de stock y escasez requerido para la optimización del inventario en la cadena de suministro es distintivamente determinado a fin de lograr el costo total mínimo en la cadena de suministro.

Venkatesan et al. (2009) desarrollan un modelo matemático para una red de cadena de suministro con los objetivos de minimizar el costo total y el tiempo de entrega de las órdenes y proponen el uso de la técnica de computación evolutiva NSGA-II, usando medidas de espacio mínimas, el desempeño del algoritmo propuesto lo compara con la RWBGA.

Lim & Xu (2005) muestran en su artículo como un algoritmo evolutivo puede jugar un efectivo rol en este aspecto. Además de ser un efectivo y eficiente herramienta de optimización capaz de manejar problemas de significativa complejidad computacional, gozando de una gran flexibilidad para hacer frente a las restricciones típicas de los problemas de la vida real en la cadena de suministro. Su trabajo considera el problema de administrar una red de distribución para la reposición de los suministros de combustible para las estaciones localizadas en toda la isla. Su metodología de solución es un EA capaz de generar soluciones aproximadas con tiempo computacional razonable.

Shu-Hsien Liao et al. (2011) proponen un modelo integrado donde incorporan decisiones de control de inventario, tales como orden económica, stock de seguridad y decisiones de reposición de inventario, dentro de los modelos típicos de localización de instalaciones, los cuales son utilizados para resolver el problema de la red de distribución. Desarrollan un algoritmo evolutivo multiobjetivo mediante la técnica NSGA-II, para determinar el óptimo entre la localización de instalaciones y los parámetros de control de inventario, con el fin de alcanzar el mejor equilibrio entre estos criterios.

Forouzanfar & Tavakkoli-Moghaddam (2012) abordan un problema de diseño de cadena de suministro multi-escalón con una solo tipo de fuente y los sistemas de inventario

relacionados, además en su modelo contemplan un nuevo modelo matemático que considera el efecto de riesgo, el tiempo de entrega e inventario multi-escalón bajo la incertidumbre de la demanda, ruta de vehículos de los centros de distribución a los clientes, con el fin de dar un servicio en un sistema de cadena de suministro estocástico, simultáneamente. El objetivo es minimizar el costo total del sistema. Utilizan el software GAMS para resolver problemas de menor tamaño. Para resolver las instancias de gran tamaño usan un algoritmo genético. Desarrollan el análisis de sensibilidad para investigar el impacto efectivo de los parámetros en las soluciones finales.

Javanshir et al. (2012) presentan una red de cadena de suministro multietapa, considerando múltiples productos, un solo recurso y costos y demanda determinísticos. Consideran dos objetivos, minimizar el costo total y minimizar el número total de productos y servicios retrasados. Su modelo propuesto es formulado como uno de programación lineal entera mixta, y para su solución presentan dos metaheurísticas llamadas MOPSO y NSGA-II. Los resultados preliminares que obtienen indican que MOPSO es capaz de generar más soluciones de Pareto en relativamente menos cantidad de tiempo, y el NSGA-II, es capaz de proveer mejor calidad de resultados.

Radhakrishnan et al. (2009) en su presentación, desarrollan un nuevo y eficiente enfoque con algoritmos genéticos, con el fin de determinar el más probable exceso en los niveles de stock y niveles de escasez, requeridos para la optimización del inventario en una cadena de suministro, tal que el costo total de la cadena de suministro es minimizado.

3. Diseño de una Cadena de Suministro con decisiones de eficiencia, Localización e Inventario usando un Algoritmo Evolutivo Multiobjetivo

Artículo presentado el 10 de octubre a las 11:00 hrs. en el XVII Congreso Latino-Iberoamericana de Investigación Operativa celebrado del 6 al 10 de octubre del 2014 en la ciudad de Monterrey Nuevo León, México.

Rodolfo Eleazar Pérez Loaiza
UPAEP, 21 Sur 1103, Puebla, Puebla 72410, México
reploaiza@gmail.com

Elías Olivares Benítez
UPAEP, 21 Sur 1103, Puebla, Puebla 72410, México
elias.olivares@upaep.mx

José H. Ablanado
The University of Texas at El Paso, 500 W. University Ave. El Paso, Texas 79968-0544, U.S.
jablanedorosas2@utep.edu

Aarón Guerrero Campanur
I. T. Superior de Uruapan, Carr. Uruapan-Carapan 5555, Uruapan, Michoacán 60015, México
agcampanur@gmail.com

José Luis Martínez Flores
UPAEP, 21 Sur 1103, Puebla, Puebla 72410, México
joseluis.martinez01@upaep.mx

Abstract

Este trabajo presenta la utilización de una metaheurística basada en un algoritmo evolutivo multiobjetivo para resolver el modelo bi-objetivo de programación no lineal entero-mixto del diseño de una cadena de suministro de tres niveles. Los proveedores envían las materias primas a un conjunto de plantas manufactureras para su procesamiento, éstas envían el producto terminado a un conjunto de centros de distribución, para que posteriormente sean reenviados a un conjunto de minoristas. Las variables de decisión son la apertura de plantas y centros de distribución, considerando una política de inventario de revisión continua, y el flujo de materiales entre las diferentes instalaciones. Los objetivos en conflicto son el de seleccionar a los proveedores en base a una métrica denominada OEE (Overall Equipment Effectiveness) que se busca maximizar y minimizar los costos totales en los que se incurre en toda la cadena. Se resuelven instancias de pequeño, mediano y gran tamaño.

Palabras Clave: Multiobjetivo; Algoritmo Evolutivo; Pareto; OEE.

1 Introducción

El diseño de la cadena de suministro consiste en determinar como elegir las entidades y distribución de bienes, para satisfacer las demandas de los clientes con un costo total mínimo (Ji & Zhao, 2006), lo que significa encontrar las respuestas a preguntas tales como: ¿Dónde podrían localizarse las plantas o centros de distribución?, ¿Qué productos podrían ser producidos y en que plantas?, por

mencionar algunas (Li & Schulze, 2011). Por lo que el éxito de las compañías está enfocado en reducir costos, los tiempos de producción, tiempos de entrega, niveles de stock, ampliar el rango de productos, tener tiempos de entrega confiable, mejorar el servicio al cliente, alta calidad, y proporcionar la eficiente coordinación entre demanda, suministros y producción (Taskin Gumus et al. 2009). Hoy en día el uso de técnicas metaheurísticas, se ha convertido en un método capaz de proporcionar rápidamente soluciones casi óptimas a problemas donde la optimización exacta está limitada, en una cantidad razonable de tiempo, realizando una búsqueda exhaustiva (Glover & A. Kochenberger, 2003). Además de manejar problemas de gran tamaño, las metaheurísticas basadas en algoritmos evolutivos, por ejemplo, generan soluciones a problemas complejos cuando las restricciones o la función objetivo no son continuas o diferenciables (Galbreth et al. 2008). El diseño de éstas metaheurísticas, son fáciles de actualizar cuando se introducen al mundo real (Stanley E. et al. 2012). Un ejemplo es el que presenta Olivares Benitez et al. (2013), donde en su publicación presenta el algoritmo de una metaheurística que combina principios de funciones voraces (greedy), búsqueda dispersa (scatter search), reencadenamiento de trayectorias (parth relinking) y programación matemática, para resolver el problema de selección de canales de transportación en el diseño de una cadena de suministro. Se presenta una metaheurística basada en un algoritmo evolutivo multiobjetivo como método de optimización aproximado para resolver el modelo bi-objetivo de programación matemática no lineal de una cadena de suministro con selección de proveedores en un problema de localización- inventario (Guerrero Campanur et al. 2013), que genere un conjunto de soluciones de los frentes de Pareto en instancias pequeñas, medianas y grandes. La complejidad del modelo radica en el conflicto que se genera entre dos funciones objetivos cuando son optimizadas simultáneamente, la selección de proveedores en base a una métrica denominada OEE (Overall Equipment Effectiveness) que busca maximizar, y los costos totales en toda la cadena que busca minimizar, además de satisfacer todas las restricciones que se generan en el problema. En las siguientes secciones se presenta la revisión de literatura, la metodología, la experimentación, un análisis de los resultados obtenidos y las conclusiones.

2 Revisión de Literatura

En la literatura de diseño de cadenas de suministro existen varios trabajos que han empleado métodos heurísticos basados en computación evolutiva en los últimos años. Radhakrishnan y Natarajan (2011), proponen un nuevo enfoque de solución basado en un algoritmo genético para resolver problemas de administración de inventarios en una cadena de suministros. Venkatesan et al. (2009), desarrollan un modelo matemático para una red de cadena de suministro con los objetivos de minimizar el costo total y el tiempo de entrega de las órdenes y proponen el uso de la técnica NSGA-II. Lim & Xu (2005) en su trabajo considera el problema de administrar una red de distribución para la reposición de los suministros de combustible para las estaciones localizadas en toda la isla. Shu-Hsien Liao et al. (2011) proponen un modelo integrado donde incorporan decisiones de control de inventario, tales como tamaño económico de la orden, stock de seguridad y decisiones de reposición de inventario, dentro de los modelos típicos de localización de instalaciones, utilizando un algoritmo evolutivo multiobjetivo basado en NSGA-II. Forouzanfar & Tavakkoli-Moghaddam (2012) abordan un problema de diseño de cadena de suministro multi-escalón con una solo tipo de fuente y los sistemas de inventario relacionados. Además, en su modelo contemplan el efecto de riesgo, el tiempo de entrega y el inventario multi-escalón bajo la incertidumbre de la demanda, y el ruteo de vehículos de los centros de distribución a los clientes, con el fin de dar un servicio en un sistema de cadena de suministro estocástico. Javanshir, Ebrahimnejad & Nouri (2012) presentan una red de cadena de suministro multietapa, considerando múltiples productos, un solo recurso y costos y demanda determinísticos. Consideran dos objetivos, minimizar el costo total y minimizar el número total de productos y servicios retrasados. Para la solución del modelo presentan dos metaheurísticas llamadas MOPSO y NSGA-II. Radhakrishnan,

Prasad & Gopalan (2009) en su presentación, desarrollan un nuevo y eficiente enfoque con algoritmos genéticos, con el fin de optimizar los niveles de inventario en una cadena de suministro.

3 Descripción del Problema y Modelo Matemático

El modelo de selección de proveedores en un problema de localización-inventario para una cadena de suministro SS-ILPSC (Guerrero Campanur et al. 2013), es un planteamiento de diseño no lineal entera mixta bi-objetivo, que puede ser considerado una extensión del problema de localización de instalación capacitado CFLP, clasificado como NP-duro. La política de inventario en una cadena de suministro de tres niveles, introducen la no linealidad en el modelo. El diseño de la cadena de suministro se compone de proveedores, plantas, centros de distribución, y los minoristas, teniendo en cuenta un solo producto, y todos los minoristas tienen una demanda y varianza independiente. Un esquema de la red de distribución se muestra en la fig.1.

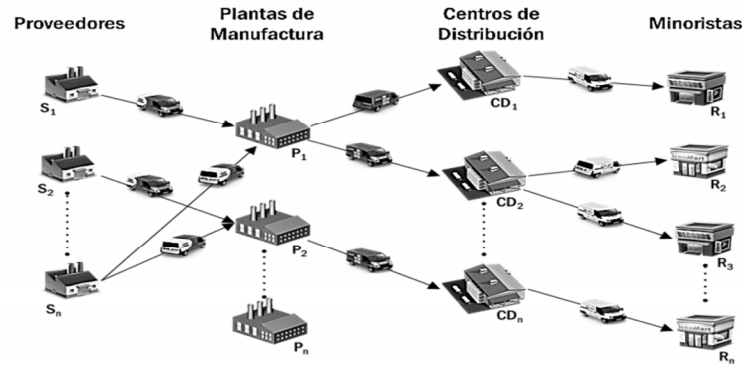


Fig. 1 Esquema de Distribución del modelo SS-ILPSC.

El modelo incorpora las operaciones tácticas dentro de las decisiones estratégicas, la selección de proveedores (basado en la OEE) y la localización-inventario, así como la determinación del flujo entre las diferentes instalaciones en cada uno de los escalones de la red, considerando la cantidad de productos en inventario con respecto a la demanda y a su varianza, mediante la política de revisión continua de inventarios, bajo un ambiente de integración de riesgo. Se deduce que la ubicación de los proveedores, plantas y centros de distribución son conocidos y tienen una capacidad. Los proveedores tienen un indicador de rendimiento - Efectividad global del equipo (OEE) que es una métrica que supervisa el rendimiento real de la instalación productiva en relación a su capacidad de funcionamiento en condiciones óptimas de operación. Se consideran dos funciones objetivas, que son optimizadas simultáneamente. La función Z_1 (1) que minimiza los costos totales que son: el costo total de inventario de trabajo, el costo del stock de seguridad, los costos fijos en los centros de distribución y plantas, los costos de transporte entre las etapas y el costo del volumen de producción. La función Z_2 (2) maximiza la efectividad global del equipo (OEE) de los proveedores a las plantas. Todo el modelo sujeto a las restricciones (3) – (16).

3.1 Variables, parámetros y notación del modelo

Conjuntos:

I	Conjunto de proveedores i	K	Conjunto de ubicaciones potenciales para cd k
J	Conjunto de ubicaciones potenciales para plantas j	L	Conjunto de clientes i

Variables de decisión:

w_{kl}	1 si el cd k sirve al cliente; 0 en otro caso.
x_k	1 si el cd k es abierto; 0 en otro caso.
y_j	1 si la planta j es abierta; 0 en otro caso.
z_{jk}	1 si la planta j sirve al cd k ; 0 en otro caso.

Variables auxiliares:

β_{ij}	Unidades producidas del proveedor i y enviadas a la planta j
δ_j	Demanda media a ser asignada a la planta j
η_j	Varianza de la demanda a ser asignada a la planta j
λ_k	Demanda media a ser asignada al cd k
π_k	Varianza de la demanda a ser asignada al cd k

Parámetros:

cs_i	Capacidad de producción del proveedor i
cw_k	Máxima capacidad del cd k
cp_j	Máxima capacidad de la planta j
d_l	Demanda media de cada cliente l
FCp_j	Costo fijo de la apertura de una planta j
FCw_k	Costo fijo de la apertura de un cd k
HC_j	Costo de mantener inventario por unidad en la planta j
HC_k	Costo de mantener inventario por unidad en el cd k
LT_{ij}	Tiempo de entrega del proveedor i a la planta j
LT_{jk}	Tiempo de entrega de la planta j al cd k
OC_{ij}	Costo por ordenar del proveedor i a la planta j
OC_{jk}	Costo por ordenar de la planta j al cd k
OEE_i	Efectividad Global del equipo para el proveedor i
A_{jk}	Costo de Transporte por unidad de la planta j al cd k
B_{kl}	Costo de transporte por unidad del cd k al cliente l
UC_i	Costo por producto producido para el proveedor i
u_l	Varianza media para cada cliente l
$Z_{1-\alpha}$	Valor de la distribución normal estándar, acumulando una probabilidad de $1-\alpha$

$$\begin{aligned} \text{Min } Z_1 = & \sum_{i=1}^I \sum_{j=1}^J \left(FCp_j \cdot y_j + \sqrt{2 \cdot OC_{ij} \cdot HC_j} \cdot \sqrt{\delta_j} + Z_{1-\alpha} \cdot HC_j \cdot \sqrt{LT_{ij}} \cdot \sqrt{\eta_j} \cdot r_{ij} + UC_i \cdot \beta_{ij} \right) + \\ & \sum_{j=1}^J \sum_{k=1}^K \left(FCw_k \cdot x_k + \sqrt{2 \cdot OC_{jk} \cdot HC_k} \cdot \sqrt{\lambda_k} + Z_{1-\alpha} \cdot HC_k \cdot \sqrt{LT_{jk}} \cdot \sqrt{\pi_k} \cdot z_{jk} + A_{jk} \cdot \lambda_k \cdot z_{jk} \right) + \\ & \sum_{k=1}^K \sum_{l=1}^L B_{kl} \cdot d_l \cdot w_{kl} \end{aligned} \quad (1)$$

$$\text{Max } Z_2 = \frac{\sum_{i=1}^I \sum_{j=1}^J (\beta_{ij} \cdot OEE_i)}{\sum_{l=1}^L d_l} \quad (2)$$

s.a.

$$\sum_{k=1}^K w_{kl} = 1 \quad \forall l = 1, \dots, L \quad (3)$$

$$\sum_{j=1}^J z_{jk} = x_k \quad \forall k = 1, \dots, K \quad (4)$$

$$\sum_{l=1}^L d_l \cdot w_{kl} = \lambda_k \quad \forall k = 1, \dots, K \quad (5)$$

$$\sum_{l=1}^L u_l \cdot w_{kl} = \pi_k \quad \forall k = 1, \dots, K \quad (6)$$

$$\sum_{k=1}^K \lambda_k \cdot z_{jk} = \delta_j \quad \forall j = 1, \dots, J \quad (7)$$

$$\sum_{k=1}^K \pi_k \cdot z_{jk} = \eta_j \quad \forall j = 1, \dots, J \quad (8)$$

$$\sum_{i=1}^I \beta_{ij} = \delta_j \quad \forall j = 1, \dots, J \quad (9)$$

$$\lambda_k \leq cw_k \cdot x_k \quad \forall k = 1, \dots, K \quad (10)$$

$$\delta_j \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (11)$$

$$\sum_{i=1}^I \beta_{ij} \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (12)$$

$$\sum_{j=1}^J \beta_{ij} \leq cs_i \quad \forall i = 1, \dots, I \quad (13)$$

$$r_{ij} = \begin{cases} 0 & \text{si } \beta_{ij} = 0 \\ 1 & \text{si } \beta_{ij} > 0 \end{cases} \quad \begin{matrix} \forall i = 1, \dots, I, \\ \forall j = 1, \dots, J \end{matrix} \quad (14)$$

$$w_{kl}, x_k, y_j, z_{jk} \in \{0, 1\} \quad \begin{matrix} \forall j = 1, \dots, J, \\ \forall k = 1, \dots, K, \\ \forall l = 1, \dots, L \end{matrix} \quad (15)$$

$$\beta_{ij} \in \mathbb{Z}^+ \quad \begin{matrix} \forall i = 1, \dots, I, \\ \forall j = 1, \dots, J \end{matrix} \quad (16)$$

4 Metodología

Este trabajo utiliza un algoritmo evolutivo multiobjetivo, basado en la técnica presentada por Carlos M. Fonseca y Peter J. Fleming, conocida como MOGA (*multiobjective genetic algorithm*). Este algoritmo usa el ranqueo no dominado (*non-domination ranking*) y un contador de nicho (*niche count*) de la población de un algoritmo genético. Se hace mucho énfasis en la no dominancia de las soluciones y simultáneamente en su diversidad (Deb, 2008). Este algoritmo difiere del algoritmo genético estándar en la manera de calcular la aptitud de cada individuo de la población, utilizando la aptitud compartida (*shared fitness*), mientras que el resto del procedimiento del algoritmo es el mismo. Se utiliza el método de selección universal estocástica (SUS), un único punto de cruce y un bit de mutación (Fonseca & Fleming, 1993). El algoritmo de muestra en la figura 3. La técnica consiste en que a cada individuo de la población se le asignará un ranking con el que será ordenado para su selección. El ranking se asigna según un criterio de no dominancia. Los individuos no dominados tendrán un ranking de 1. Los individuos dominados son penalizados de acuerdo a la densidad de las regiones que lo dominan. Si x_i es no dominado entonces el ranking de $x_i = 1$, en otro caso el ranking de $x_i = 1 +$ (número de individuos que lo dominan) (Coello Coello et al. 2007), es decir, el ranking de un cierto individuo corresponde al número de individuos en la actual población por los cuales está dominado. Se calcula la aptitud promedio (average fitness), el contador de nicho (*niche count*), la aptitud compartida (*shared fitness*) y el valor de escalamiento de la aptitud compartida (scaled shared fitness) de cada individuo, para mantener la diversidad entre las soluciones no dominadas, para mayor referencia sobre el uso de las funciones aquí mencionadas, consultar a Deb Posteriormente se realiza la selección de los individuos que serán considerados para tener descendencia mediante el método de Muestreo Universal Estocástico (SUS). Para el proceso del cruzamiento se considera una probabilidad del 90% y para la mutación una probabilidad del 1%. Antes de finalizar cada generación se seleccionan a los mejores individuos con ranking igual a 1 de toda la población y son unificados con los obtenidos en una generación anterior, creando un solo conjunto de individuos con ranking igual a 1. En seguida a cada individuo de este grupo se le reasigna ranking de acuerdo al criterio de no dominancia para que posteriormente se filtren aquellos mejores con ranking igual a 1, que formarán el frente de Pareto de esa generación y que serán preservados. El proceso se repite en cada iteración hasta finalizar el algoritmo. Este frente de Pareto que se actualiza en cada generación es el producto final del algoritmo.

Algoritmo Evolutivo Multiobjetivo

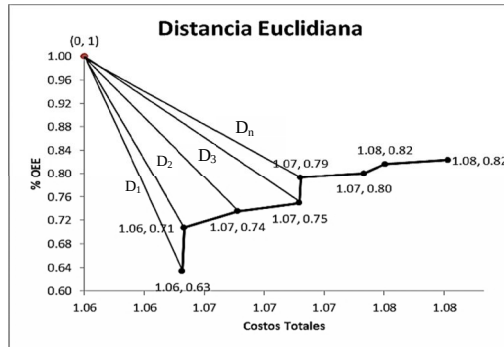
INICIO			
Entrada: Datos de la instancia del modelo SS-ILPSC		12.	Cruza en un solo punto
Salida: Conjunto de soluciones no dominadas.		13.	Mutación
1.	Genera población inicial (<i>IP</i>)	14.	Evaluar funciones objetivo (<i>EOF</i>)
2.	Evaluar funciones objetivo (<i>EOF</i>)	15.	Asignar Ranking Basado en Dominancia de Pareto (<i>Rank</i>)
3.	Asignar Rango Basado en Dominancia de Pareto (<i>Rank</i>)	16.	Calcular la aptitud promedio (<i>AF</i>)
4.	Calcular la aptitud promedio (<i>AF</i>)	17.	Calcular el conteo de nicho (<i>NC</i>)
5.	Calcular el conteo de nicho (<i>NC</i>)	18.	Calcular aptitud compartida (<i>SF</i>)
6.	Calcular aptitud compartida (<i>SF</i>)	19.	Escalar la aptitud compartida (<i>SCF</i>)
7.	Escalar la aptitud compartida (<i>SCF</i>)	20.	Seleccionar a los mejores con ranking 1 (<i>B2</i>)
8.	Seleccionar a los mejores con ranking 1 (<i>B1</i>)	21.	Unir los mejores de esta generación con la anterior (<i>B1B2</i>)
9.	Generar Frente de Pareto	22.	Asignar Rango Basado en Dominancia de Pareto (<i>Rank</i>)
10.	Para (<i>i</i> =1 hasta <i>N</i> Generaciones)	23.	Elegir a los mejores con ranking 1 (<i>BOB</i>)
11.	Selección por Muestreo Universal Estocástico (<i>SUS</i>)	24.	Generar Frente de Pareto
		25.	Fin de Ciclo
FIN			

Fig. 3. Algoritmo Evolutivo Multiobjetivo

5 Experimentación y Resultados

Siguiendo cada uno de los pasos de acuerdo al procedimiento del algoritmo evolutivo multiobjetivo propuesto, se establecen los siguientes objetivos para evaluar su eficacia, eficiencia y desempeño computacional. Lo primero es valorar el funcionamiento con instancias de diferentes tamaños para observar el número de soluciones de cada frente de Pareto que se generan en cada una de ellas y en que tiempo. La segunda meta es presentar un estudio detallado del mejor frente de Pareto obtenido mediante un análisis estadístico de los resultados que se obtienen. Lo que se busca es que la persona encargada de tomar las decisiones en su análisis elija la solución que mejor le convenga, considerando los objetivos planteados previamente. Para probar el desempeño computacional, las instancias son generadas de acuerdo al código formado por cuatro parámetros: el número de proveedores-el número de plantas-el número de centros de distribución-el número de clientes, como se muestra en la columna 1, de la tabla 1.

Los resultados que se presentan son obtenidos considerando los siguientes parámetros en el algoritmo evolutivo para cada instancia: se considera dos valores para el tamaño de población de 200 y 600, el total de generaciones con un valor de 100, el porcentaje de cruce del 90% y el de mutación del 1%. Se produjeron 10 réplicas para cada tamaño de la población en cada una de las diferentes instancias. Para poder comparar las réplicas generadas con el frente de Pareto real, se hace el uso de la distancia euclidiana con la ecuación (24). El proceso consiste en determinar que distancia existe desde un punto de referencia hacia cada uno de los puntos del frente de Pareto obtenido en cada réplica. En este caso se está considerando la coordenada (0,1) como punto de referencia, que es el punto donde la función objetivo correspondiente a los costos totales tiene un valor de 0 y el punto máximo que puede obtener la función objetivo correspondiente al índice del OEE es 1. En la figura 2 se muestra un ejemplo de como se realiza este procedimiento. Una vez calculada la distancia de cada punto se determina el promedio de todas las distancias, obteniendo así el valor de la media de cada frente de Pareto, el cual posteriormente es comparado con el de los demás de cada réplica, en cada instancia.



$$d_E(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (24)$$

Fig. 2. Frente de Pareto usando distancia euclidiana

Para medir el desempeño del algoritmo evolutivo, se consideraron instancias pequeñas como grandes, que generen conjuntos de soluciones no dominadas aproximadas al frente de Pareto real. Todos los procedimientos fueron codificados en Matlab versión 7.12.0.635, en una PC HP 6730s, Intel® Core™2 Duo, 2.00GHz y 2.0 GB en RAM. Las instancias de los grupos 5-3-5-10, 5-3-10-10, 5-3-10-15, 5-5-5-15 y 5-5-10-20, fueron resueltas completamente con el algoritmo evolutivo y los resultados se muestran en la tabla 1. Los resultados incluyen el número de soluciones, el tiempo derivado en segundos y la distancia euclidiana promedio de cada réplica, así como la media y la desviación estándar. Se puede observar que en el 100% de todos los casos, se obtienen mejores resultados al correr el algoritmo evolutivo considerando una población mayor que una de menor

cantidad, durante un periodo de 100 generaciones, como se muestra en la tabla 1. Es importante mencionar que no se considera otro valor en este parámetro, debido a que el conjunto de soluciones no dominadas de cada uno de los frentes obtenidos, ya no presenta cambios mucho antes de que finalice la generación número 100. Una principal característica de los frentes obtenidos es que son discontinuos, esto es por la naturaleza discreta del problema.

Tabla 1. Comparación de réplicas en términos de su media y desviación estándar.

Instancia	P	Réplicas										Media	Desv.Est.
		1	2	3	4	5	6	7	8	9	10		
5-3-5-10	200	S	4	3	4	3	3	3	3	6	6	1.05	0.013
		T	26.71	26.94	25.15	25.34	28.14	26.89	24.51	25.40	24.97		
		D	1.06	1.05	1.04	1.05	1.04	1.05	1.04	1.06	1.08		
	600	S	4	3	3	3	3	3	3	3	3	1.04	0.007
		T	80.82	75.43	75.09	74.38	73.82	78.79	74.78	74.08	74.08		
		D	1.06	1.04	1.04	1.04	1.04	1.05	1.04	1.04	1.04		
5-3-10-10	200	S	14	16	13	18	15	14	15	13	14	1.12	0.014
		T	45.96	45.76	46.45	46.13	46.43	45.41	46.05	45.90	45.76		
		D	1.11	1.13	1.12	1.10	1.12	1.12	1.12	1.14	1.13		
	600	S	14	15	15	16	14	9	17	16	15	1.11	0.006
		T	139.02	138.01	137.45	137.88	140.22	138.00	137.83	138.38	137.64		
		D	1.10	1.11	1.11	1.12	1.11	1.12	1.12	1.11	1.11		
5-3-10-15	200	S	6	8	6	5	7	8	8	5	7	1.42	0.010
		T	39.41	34.76	55.63	34.07	52.20	34.55	35.00	34.87	35.53		
		D	1.43	1.42	1.40	1.42	1.41	1.43	1.43	1.43	1.42		
	600	S	4	8	8	5	6	6	7	7	5	1.41	0.010
		T	104.94	104.43	106.94	106.38	106.47	104.96	104.32	106.05	139.01		
		D	1.41	1.42	1.42	1.42	1.41	1.42	1.41	1.41	1.39		
5-5-5-15	200	S	6	3	3	4	3	5	5	3	5	1.43	0.007
		T	33.09	34.06	33.00	33.71	32.80	32.95	33.09	32.90	33.04		
		D	1.43	1.42	1.42	1.43	1.43	1.43	1.43	1.42	1.43		
	600	S	3	4	9	5	9	9	7	9	4	1.41	0.008
		T	98.01	100.45	100.24	98.92	98.66	99.05	98.20	98.67	98.36		
		D	1.42	1.42	1.40	1.42	1.42	1.42	1.40	1.40	1.42		
5-5-10-20	200	S	6	6	6	5	7	5	5	6	5	2.03	0.030
		T	43.34	41.50	41.87	40.88	42.38	41.01	41.31	41.69	41.43		
		D	2.02	2.03	2.00	2.07	2.00	2.05	2.01	2.00	2.01		
	600	S	6	7	6	6	8	5	5	5	7	1.99	0.017
		T	123.77	124.14	124.16	124.53	125.29	125.28	124.59	124.91	125.21		
		D	1.97	1.98	2.02	2.00	2.01	2.00	1.97	1.98	1.97		

P=población S=soluciones T=tiempo (seg) D=distancia promedio

6 Conclusiones

Este trabajo presenta resultados para un hipotético pero realista problema de diseño de cadena de suministro, donde los conjuntos de soluciones generadas son presentadas para el tomador de decisiones en términos de equilibrio de costo-eficiencia. El potencial para resolver problemas combinatorios grandes mediante el uso de algoritmos evolutivos es alto, sobretodo cuando los casos son multiobjetivo. Para trabajos futuros relacionados con el modelo se puede introducir una demanda estocástica, que es una importante característica que requiere ser implementada en el diseño de una cadena de suministro, considerando que los escenarios de la vida real se comportan de esta manera. También puede ser considerado el uso de flujos directos de plantas a clientes y el caso de múltiples productos. Cuatro diferentes instancias fueron probadas para obtener el conjunto de soluciones de los frentes de Pareto. Una de las principales ventajas del algoritmo evolutivo es el tiempo de ejecución, que es relativamente corto con respecto a la complejidad y naturaleza del problema, que esta representado por un modelo no lineal. Este método de solución es una buena alternativa para instancias de pequeño, mediano y gran tamaño.

7 Reconocimientos

Este trabajo es el resultado de los estudios doctorales que se están desarrollando en la Universidad Popular Autónoma del Estado de Puebla (UPAEP), y que es sustentada por una beca otorgada por el Consejo Nacional de Ciencia y Tecnología (CONACYT).

Referencias

1. Coello Coello, Carlos A., Gary B. Lamont, y David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, Nueva York, 2007.
2. Deb, Kalyanmoy. *Multi-Objective Optimization using Evolutionary Algorithm*. John Wiley & Sons, Ltd., United Kingdom, 2008.
3. Fonseca, Carlos M., y Peter J. Fleming. «Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization.» *Genetic Algorithms: Proceedings of the Fifth International Conference.*, pp. 416-423, S. Forrest, ed., San Mateo CA: Morgan Kaufmann, 1993.
4. Forouzanfar, Fatemeh, y Reza Tavakkoli-Moghaddam. «Using a genetic algorithm to optimize the total cost for a location-routing-inventory problem in a supply chain with risk pooling.» *Journal of Applied Operational Research*, 4(1):2-13, 2012.
5. Galbreth, Michael R., James A. Hill, y Sean Handley . «An investigation of the value of cross-docking for supply Chain Management.» *Journal of Business logistics*, 29(1):225-239, 2008.
6. Glover, Fred, y Gary A. Kochenberger. *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers, 2003.
7. Guerreo Campanur, Aaron, Elías Olivares Benítez, y Pablo A. Miranda. «Supplier Selection in an Inventory Location Problem for a three-level Supply Chain.» Reporte Técnico, UPAEP, Puebla, México, 2013.
8. Javanshir, Hassan, Sadoullah Ebrahimnejad, y Samaneh Nouri. «Bi-objective supply chain problem using MOPSO and NSGA-II.» *International Journal of Engineering Computations*, pp. 681-694, 2012.
9. Ji, X, y X. Zhao. «Supply Chain Network Design under Uncertainty.» *11th Annual Conference of Asia Pacific Decision Sciences Institute*, pp. 526-529, 2006:.
10. Li, Li, y Lothar Schulze. «Uncertainty in Logistics Network Design: A review.» *Proceeding of the International Multiconference of Engineers and Computer Scientist*. 2(1), IMECS, Hong Kong, 2011.
11. Lim, M.H., y Y.L. Xu. «Application of Evolutionary Algorithm in Supply Chain Management.» *International Journal of Computers, Systems and Signals*, 6: 64-77, 2005.
12. Olivares Benitez, Elias, Roger Z. Ríos Mercado, y José Luis González Velarde. «A metaheuristic algorithm to solve the selection of transportation channels in supply chain design.» *International Journal of Production Economics*, ELSEVIER, 2013.
13. Radhakrishnan, Perumalsamy y Natarajan, Jeyanthi. «Genetic Algorithm Model for Multi-factory Supply Chain Inventory Optimization involving Lead Time.» *IJCEM International Journal of Computational Engineering & Management*, 14:142-150, 2011.
14. Radhakrishnan, P., V.M. Prasad, y M.R. Gopalan. «Inventory Optimization in Supply Chain Management using Genetic Algorithm.» *International Journal of computer Science and Network Security*, 9(1):33-40, 2009.
15. Shu-Hsien Liao, Chia-Lin Hsieh, y Yu-Siang Lin. «A multi-objective evolutionary optimization approach for an integrated location-inventory distribution network problem under vendor-managed inventory systems.» *Annals of Operations Research*, 186(1): 213-229, 2011.
16. Stanley E., Griffis, John E. Bell, y David J. Closs. «Metaheuristics in Logistics and Supply Chain Management.» *Journal of Business Logistics*, 33(2):90-106, 2012.
17. Taskin Gumus, Alev, Ali Fuat Guneri, y Selcan Keles. «Supply chain network design using an integrated neuro-fuzzy and MILP approach: A comparative design study.» *Expert Systems with Applications*, pp.12570-12577. ELSEVIER, 2009.
18. Venkatesan, S Prasanna, S. Kumanan, y S. Karthik. «Multi-criteria decision making in a supply chain using a genetic algorithm.» *ICFAI Journal of Supply Chain Management*. 6(2):67-82, 2009.

4. Supply Chain Design with Efficiency, Location and Inventory Policy using a Multi-Objective Evolutionary Algorithm

Abstract

This study presents a metaheuristic based on a multi-objective evolutionary algorithm to solve a bi-objective mixed-integer nonlinear programming model for the supply chain design with location-inventory decisions and supplier selection. In the model, suppliers send raw materials to a set of manufacturing plants for processing. These send the finished product to a set of distribution centres which then send them to a set of retailers. The decision variables are the opening of plants and distribution centres, considering a continuous review inventory policy and the flow of materials between the different facilities. The conflicting objectives are: to minimise total costs on the entire chain, and the selection of suppliers with the highest combined value with regard to a metric called OEE (Overall Equipment Effectiveness). Small and medium scale scenarios are solved. The results are compared with optimal Pareto fronts obtained using commercial optimisation software that applies the epsilon-constraint method. The numerical results show the effectiveness of the proposed metaheuristic.

Key Words: Multi-objective; Evolutionary Algorithm; Pareto Front; OEE.

1 Introduction

Organisations today exhibit very dynamic behaviour in their attempts to adapt to globalised markets that are constantly changing, offering a high variety of products at considerable low cost. Companies must constantly observe the processes involved in their supply chains, focusing on finding solutions that decrease production time, delivery time, inventory levels and costs in general, while at the same time improving the quality of their products and customer service, and establishing efficient coordination between demand, supply and production (Taskin Gumus et al., 2009).

Supply chain design is an important strategic decision problem, involving questions such as: Where should production plants or distribution centres be located? Which products should be produced and at which plants? How will goods be distributed between the different entities? (Li & Schulze, 2011) In order to answer these questions it is necessary to use models that meet the demands of customers at minimum cost (Ji & Zhao, 2006). One example is the simultaneous modelling of facility locations, inventory control, fleet design and customers grouping (Miranda et al., 2009). This leads to the search for solution methods that solve these types of problems, taking into account a considerable number of decision variables and constraints that significantly affect the performance of the entire supply chain of an organisation. Due to the size, complexity and often the nonlinear nature of these problems, such as the problem analysed in this study, a closed approach produced by an exact method for solving the problem is not always available. Nowadays, the use of metaheuristics has become capable of providing rapid and almost optimal solutions to problems in a reasonable amount of time when exact optimisation is limited by conducting an exhaustive search (Glover & Kochenberger, 2003). One example is the study by Miranda & Garrido (2004) on a mixed-integer nonlinear model and a solution heuristic based on Lagrangian relaxation applied to the inventory control decision problem together with the distribution network design, obtaining satisfactory results. Purnomo et al. (2012) model different inventory policies in a supply chain

configuration and design problem, using harmony search as the solution method. In another example, Olivares Benitez et al. (2013) present the algorithm of a metaheuristic that combines principles of greedy functions, scatter search, path relinking and mathematical programming to solve the transportation channel selection problem in supply chain design.

Some of the most well-known and commonly used metaheuristics in recent years are evolutionary algorithms (EA). These have led to considerable advancements when used in large and complex problems. The approach has the advantage of providing feasible solutions that are close to optimal even when the constraints or the objective function of the problem are not continuous, and thus do not require a linear formulation. Their ability is to search simultaneously in different regions of the solution space, making it possible to find a set of different solutions (Konak et al., 2006). This ability is important when dealing with supply chain design problems as it gives the user different solution options, considering limitations or constraints and conditions that can frequently change (cost, distance, speed, etc.). The structure of their design, as with many other metaheuristics, allows easy updating and execution when changes occur in the real world, allowing the user to identify the advantages and disadvantages in objectives such as customer service or logistics costs.

The present study looks at the supplier selection problem within a location problem with inventory for a supply chain. It is assumed that the number of suppliers and customers is known. The plants and distribution centres are not defined, and only potential sites are known. Each possible plant and distribution centre has assigned a fixed cost and a capacity limit (Guerrero, 2013). The complexity of the model lies in the conflict between two objective functions when they are optimised simultaneously. One aims to decrease total costs for the entire chain and the other aims to select the suppliers with highest value on a metric variable called OEE (Overall Equipment Effectiveness). The solution must also satisfy all the constraints arising from the problem. This study presents the metaheuristic of a multi-objective evolutionary algorithm based on the technique known as MOGA (multi-objective genetic algorithm) (Fonseca & Fleming, 1993) to solve the aforementioned problem. A comparison is made between the results obtained with the evolutionary algorithm and the optimal Pareto fronts obtained with the commercial optimisation software LINGO ver. 14.0 applying the epsilon-constraint method for small and medium-scale scenarios. The sections below contain a review of the literature, the methodology, the experimentation, and analysis of the results obtained and the conclusions.

2 Literature Review

The literature contains several studies that have employed heuristic methods to find solutions to certain supply chain design problems, specifically evolutionary computation, due to the significant achievements that have been seen in recent years. Radhakrishnan and Natarajan (2011) propose a new approach based on a genetic algorithm to solve inventory management problems on a supply chain. Venkatesan et al. (2009) developed a mathematical model for a supply chain network with the objective of minimising total cost and order delivery time, proposing the use of the evolutionary computation technique NSGA-II (Non-Dominated Sorting Genetic Algorithm). Lim & Xu (2005) show how an evolutionary algorithm can play an effective role in this aspect. Their study considers the problem of managing a distribution network for replacing fuel supplies for fuel stations located throughout an island. Shu-Hsien Liao et al. (2011) propose an integrated model which incorporates inventory control decisions, such as economic order quantity, safety stock and inventory repositioning decisions, within typical facility location models, using a multi-objective evolutionary algorithm based on NSGA-II. Forouzanfar & Tavakkoli-Moghaddam (2012) address a multi-echelon supply chain design problem with a single type of source and the associated inventory systems. Their model also includes the effect of risk, delivery time and multi-echelon inventory

under demand uncertainty, and vehicle routing from distribution centres to customers, with the aim of providing service in a stochastic supply chain system. Javanshir, Ebrahimnejad & Nouri (2012) present a multi-echelon supply chain network, considering multiple products, a single resource and deterministic costs and demand. They consider two objectives, minimising total costs and minimising the total number of delayed products and services. They present two metaheuristics to solve the problem, MOPSO (Multi-objective Particle Swarm Optimisation) and NSGA- II. Radhakrishnan, Prasad & Gopalan (2009) develop a new and efficient approach using genetic algorithms with the aim of optimising inventory levels on a supply chain.

Kim et al. (2012) present a selection algorithm for non-dominated solutions obtained by a MOEA, in accordance with a degree of consideration that represents user preferences. Wang & Cai (2012) developed an improved version of the CW method (Cai Wang method), called CMODE (Combined Multi-objective Optimisation with Differential Evolution) to consider multi-objective optimisation problems with constraints. Thu Bui et al. (2012) present a review of the use of evolutionary algorithms for solving optimisation problems in dynamic scenarios, and propose a computational approach for multi-objective optimisation for dynamic planning problems, adapting the current solution when changes occur. Kleeman et al. (2007) develop the design of a multi-objective evolutionary algorithm to solve a variation problem in the design of a communication network, based on NSGA-II. Sindhya, Miettinen & Deb (2013) propose the framework for a hybrid multi-objective evolutionary algorithm with a modular structure that can be used to implement hybrid multi-objective optimisation algorithms. Xing et al. (2010) use evolutionary algorithms to find solutions to the vehicle routing problem. Karahan & Köksalan (2010) developed a steady-state elitist evolutionary algorithm to approximate optimal Pareto fronts for multi-objective decision-making problems.

Hnaïen et al. (2010) consider two multi-objective approaches based on genetic algorithms which are evaluated with a variety of supply chain configurations. Validi et al. (2014) propose an effective method of solving a two-echelon supply chain model using MOGA-II. Miandoabchi et al. (2013) present an urban road network design as part of a multi-objective decision-making model using a hybrid genetic algorithm, a known evolutionary simulation and a hybrid artificial bee colony algorithm. Latha Shankar et al. (2013) performed multi-objective optimisation of a four-echelon supply chain using a multi-objective hybrid particle swarm optimisation algorithm (MOHPSO) incorporating a non-dominated sorting algorithm. Altıparmak et al. (2006) propose a new solution procedure based on genetic algorithms for finding a set of approximate Pareto solutions for a multi-objective supply chain design problem, with the implementation of two different weighting methods. Latha Shankar et al. (2013) solve a three-echelon supply chain using a hybrid particle swarm optimisation algorithm that incorporates a non-dominated sorting (NDS) procedure to improve the bi-objective optimisation.

3 Description of the Problem and Mathematical Model

The integrated model of supplier selection within an inventory-location problem for a supply chain (Guerrero, 2013) has a bi-objective mixed-integer nonlinear structure that can be considered an extension of the Capacitated Facility Location Problem (CFLP), classified as NP-hard. The inventory policy in a three-echelon supply chain introduces nonlinearity into the model. The supply chain is comprised of suppliers, manufacturing plants, distribution centres and retailers, considering a single product and that all retailers have independent variance and demand. A scheme of this distribution network is shown in fig. 1.

The model tactical and strategic decisions: supplier selection, plant and distribution centre location and product flow between the different facilities in each step of the network, taking into account inventory decisions and costs as a function of expected demand and its variance, considering a continuous review inventory policy with risk pooling. It is assumed that the location of the suppliers is known. There is a set of potential plants and distribution centres with specific capacities. The suppliers have a performance index, OEE (Overall Equipment Effectiveness), for their selection. This is a metric that values the real performance of a production facility in relation to its working capacity under optimal conditions.

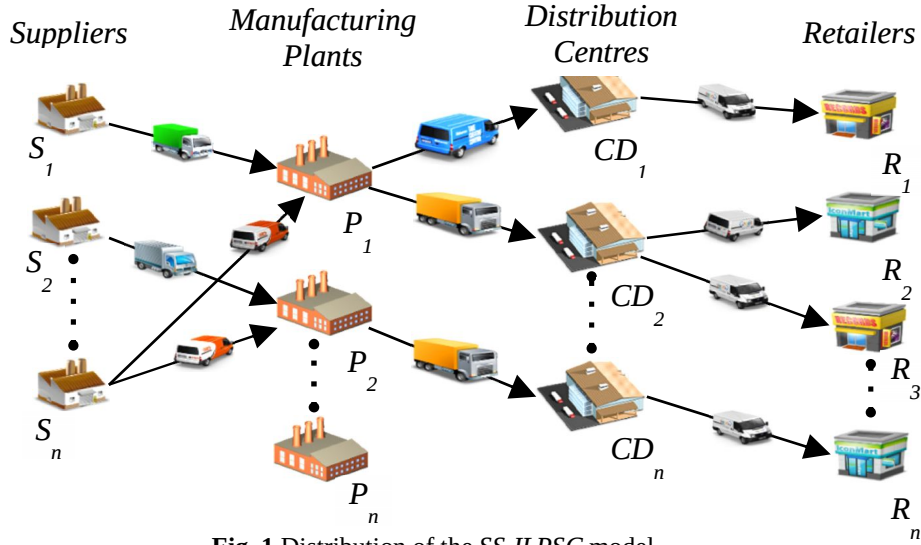


Fig. 1 Distribution of the SS-ILPSC model.

In this case, OEE includes three fundamental parameters of industrial production: availability of facilities to carry out production, the efficiency with which products are manufactured and the quality of the products in compliance with necessary standards; it is calculated using equation (1). It is important to note that this is only a generalised example of how a metric is used; the idea can be applied to other types of indicators to measure supplier performance.

$$OEE_i = \left(\frac{et_i}{st_i} \right) \cdot \left(\frac{ct_i \cdot tp_i}{et_i} \right) \cdot \left(\frac{tp_i - td_i}{tp_i} \right) \quad (1)$$

In equation (1) et_i is the actual production time of supplier i , st_i is the planned production time of supplier i , ct_i is the ideal cycle time of supplier i , tp_i is the total number of products produced by supplier i , and td_i is the total number of defective products from supplier i .

3.1 Model variables, parameters and annotation

The formulation of the mathematical model introduces the following notation.

Sets:

- I Set of suppliers i .
- J Set of potential locations for plants j .
- K Set of potential locations for distribution centres k .
- L Set of customers l .

Decision variables:

w_{kl}	1 if distribution centre k serves customer l ; 0 otherwise.
x_k	1 if distribution centre k is open; 0 otherwise.
y_j	1 if plant j is open; 0 otherwise.
z_{jk}	1 if plant j serves distribution centre k ; 0 otherwise.
β_{ij}	Number of units of the product produced by supplier i and sent to plant j

Auxiliary variables:

δ_j	Average demand assigned to plant j
η_j	Variance in demand assigned to plant j
λ_k	Average demand assigned to distribution centre k
π_k	Variance in demand assigned to distribution centre k
r_{ij}	Binary variable indicating if supplier i serves plant j , in which case it is set to 1. It is 0 otherwise.

$$r_{ij} = \begin{cases} 0 & \text{si } \beta_{ij} = 0 \\ 1 & \text{si } \beta_{ij} > 0 \end{cases} \quad \begin{matrix} \forall i = 1, \dots, I, \\ \forall j = 1, \dots, J \end{matrix} \quad (2)$$

Parameters:

cs_i	Production capacity of supplier i
cw_k	Maximum capacity of distribution centre k
cp_j	Maximum capacity of plant j
d_l	Average demand of each customer l
FCp_j	Fixed cost of opening plant j
FCw_k	Fixed cost of opening distribution centre k
HC_j	Cost per unit of keeping inventory at plant j
HC_k	Cost per unit of keeping inventory at distribution centre k
LT_{ij}	Delivery time of supplier i to plant j
LT_{jk}	Delivery time from plant j to distribution centre k
LT_j	Maximum delivery time of selected suppliers to plant j
OC_{ij}	Ordering Cost from supplier i to plant j
OC_{jk}	Ordering Cost from plant j to distribution centre k
OEE_i	Overall Equipment Effectiveness of supplier i
A_{jk}	Transport Cost per unit from plant j to distribution centre k
B_{kl}	Transport Cost per unit from distribution centre k to customer l
UC_i	Cost per unit of product produced by supplier i
u_l	Demand variance of each customer l
$Z_{1-\alpha}$	Value of the accumulated standard normal distribution with a probability of $1-\alpha$

The mathematical formulation is described below. The model considers two objective functions that are optimised simultaneously. Function Z_1 (3) minimises total costs, which include: total cost of working inventory, cost of safety stock, fixed costs of distribution centres and plants, transport costs between echelons and production cost .

$$\begin{aligned}
Min Z_1 = & \sum_{i=1}^I \sum_{j=1}^J \left(FCp_j \cdot y_j + \sqrt{2 \cdot OC_{ij} \cdot HC_j} \cdot \sqrt{\delta_j} + Z_{1-\alpha} \cdot HC_j \cdot \sqrt{LT_j} \cdot \sqrt{\eta_j} + UC_i \cdot \beta_{ij} \right) + \\
& \sum_{j=1}^J \sum_{k=1}^K \left(FCw_k \cdot x_k + \sqrt{2 \cdot OC_{jk} \cdot HC_k} \cdot \sqrt{\lambda_k} + Z_{1-\alpha} \cdot HC_k \cdot \sqrt{LT_{jk}} \cdot \sqrt{\pi_k} \cdot z_{jk} + A_{jk} \cdot \lambda_k \cdot z_{jk} \right) + \\
& \sum_{k=1}^K \sum_{l=1}^L B_{kl} \cdot d_l \cdot w_{kl}
\end{aligned} \tag{3}$$

Function Z_2 (4) maximises the observed OEE of suppliers to the plants.

$$Max Z_2 = \sum_{i=1}^I \sum_{j=1}^J \frac{(\beta_{ij} \cdot OEE_i)}{\sum_{l=1}^L d_l} \tag{4}$$

The entire model is subject to constraints (5) – (18). Constraint (5) ensures that each retailer is assigned to only one warehouse, and constraint (6) ensures that each warehouse is assigned to only one plant, as long as such plant has been installed ($x_k=1$).

$$\sum_{k=1}^K w_{kl} = 1 \quad \forall l = 1, \dots, L \tag{5}$$

$$\sum_{j=1}^J z_{jk} = x_k \quad \forall k = 1, \dots, K \tag{6}$$

Constraints (7) and (8) add the average demand and variance of the retails to each warehouse.

$$\sum_{l=1}^L d_l \cdot w_{kl} = \lambda_k \quad \forall k = 1, \dots, K \tag{7}$$

$$\sum_{l=1}^L u_l \cdot w_{kl} = \pi_k \quad \forall k = 1, \dots, K \tag{8}$$

Similarly, constraints (9) and (10) add the average demand and variance of the warehouses to each plant, respectively.

$$\sum_{k=1}^K \lambda_k \cdot z_{jk} = \delta_j \quad \forall j = 1, \dots, J \tag{9}$$

$$\sum_{k=1}^K \pi_k \cdot z_{jk} = \eta_j \quad \forall j = 1, \dots, J \tag{10}$$

Constraint (11) ensures that the total amount of product produced by one or more suppliers and sent to each plant is equal to the demand of each plant.

$$\sum_{i=1}^I \beta_{ij} = \delta_j \quad \forall j = 1, \dots, J \tag{11}$$

The warehouse and plant capacity constraints are given by (12) and (13). Constraints (12) and (13) ensure that the demand of each warehouse or plant is less than or equal to its respective capacity.

$$\lambda_k \leq cw_k \cdot x_k \quad \forall k = 1, \dots, K \quad (12)$$

$$\delta_j \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (13)$$

Constraint (14) establishes that the total amount of product produced by one or more suppliers and sent to each plant is less than or equal to the capacity of that plant.

$$\sum_{i=1}^I \beta_{ij} \leq cp_j \cdot y_j \quad \forall j = 1, \dots, J \quad (14)$$

Constraint (15) ensures that the total amount of product produced by supplier i and sent to one or more plants is less than or equal to the production capacity of that supplier.

$$\sum_{j=1}^J \beta_{ij} \leq cs_i \quad \forall i = 1, \dots, I \quad (15)$$

Constraint (16) calculates the maximum delivery time in the case that several suppliers are sending products to a single plant

$$LT_j \geq LT_{ij} \cdot r_{ij} \quad \forall j = 1, \dots, J \quad (16)$$

Finally, constraints (17) and (18) show the nature of the variables.

$$\begin{aligned} w_{kl}, x_k, y_j, z_{jk} &\in \{0, 1\} & \forall j = 1, \dots, J, \\ & & \forall k = 1, \dots, K, \\ & & \forall l = 1, \dots, L \end{aligned} \quad (17)$$

$$\begin{aligned} \beta_{ij} &\in \mathbb{Z}^+ & \forall i = 1, \dots, I, \\ & & \forall j = 1, \dots, J \end{aligned} \quad (18)$$

4 MOEA approach and epsilon-constraint method using LINGO

4.1 MOEA Method

The multi-objective evolutionary algorithm used in this study is based on the technique presented by Fonseca and Fleming (1993), known as MOGA (multi-objective genetic algorithm) which simultaneously uses Pareto ranking and the niche count technique. Its aim is to search for the true and complete Pareto front while maintaining diversity in the population, which is an important consideration in a multi-objective evolutionary algorithm in order to obtain uniformly distributed solutions that cover the Pareto front and avoids the phenomenon known as genetic deviation (Konak et al., 2006).

4.1.1 Solution representation

Since the supply chain design in this model contemplates three echelons, the coding of each chromosome is the assignment of a possible combination of integers that represent the flow of materials from suppliers to retailers in a chain of matrices that represent each of the different echelons. Therefore, an allele of the chromosome is an echelon on the chain, as shown in figure 2.

The plants that are chosen to be supplied by the suppliers will in turn provide products to the chosen distribution centres to be divided between the different retailers considering all the constraints in the model. The dimension of each matrix is in relation to the number of suppliers, plants, distribution centres and retailers.

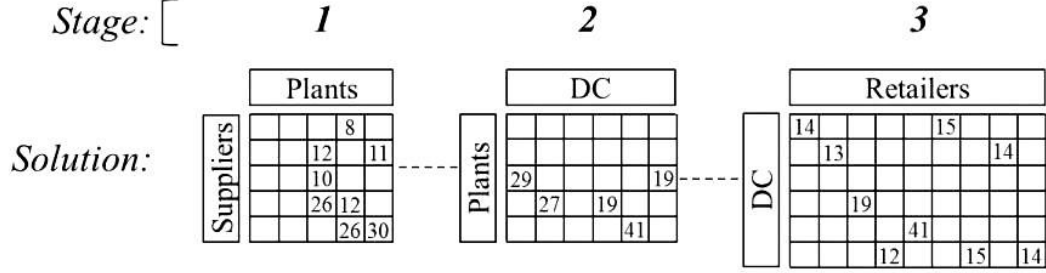


Fig. 2 Solution representation.

As well as allowing a practical representation of the different combinations that are possible on the entire supply chain, this representation also facilitates the configuration of genetic operations to manipulate chromosomes during the evolutionary search process.

4.1.2 Proposed Pareto Ranking

In Pareto ranking, each solution i of the population is ranked according to a dominance rule for evaluating fitness, leading to the order used for selection (Deb, 2008). If solution i is not dominated then its ranking $r(i) = 1$, which means that all non-dominated solutions will have a ranking of 1, otherwise the ranking is $r(i) = 1 + n$, where n = number of solutions in the current population which dominate solution i . Therefore, solutions that are dominated are penalised according to the number of solutions that dominate them, and thus a lower ranking corresponds to a better solution (Coello Coello et al., 2007). Once the ranking has been calculated for each solution, this is assigned as a first fitness, giving a first indication of which solutions are the best.

4.1.3 Preservation of Diversity

In order to maintain diversity among the non-dominated solutions, the niche count technique is introduced to penalise solutions grouped together with the same ranking, using a fitness sharing function to obtain an estimate of solutions belonging to each ranking as follows (Fonseca & Fleming, 1993):

Step 1: The first fitness of the solutions of each ranking are averaged using equation (19) to ensure that those with a better ranking are assigned a higher fitness.

$$F_i' = M - \sum_{k=1}^{r_i-1} \mu r_k - 0.5(\mu r_i - 1) \quad (19)$$

Where M is the number of solutions in the population, μr_k is the number of solutions with rank k and μr_i is the number of solutions with rank r_i .

Step 2: The normalised distance (d_{ij}) between each pair of solutions of the objective function with the same ranking is calculated using equation (20)

$$d_{ij} = \sqrt{\sum_{k=1}^M \left(\frac{f_k^{(i)} - f_k^{(j)}}{f_k^{\max} - f_k^{\min}} \right)^2} \quad (20)$$

Where $f_k^{\max}$ and $f_k^{\min}$ are the maximum and minimum values of the objective function f_k , M is the number of solutions in the population.

Step 3: With the distances obtained in step 2, the fitness sharing function, $Sh(d_{ij})$, is calculated using equation (21) to obtain the niche count, nc_i , for each solution with the same ranking using equation (22).

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}} \right)^\alpha, & \text{if } d \leq \sigma_{share}; \\ 0, & \text{otro caso.} \end{cases} \quad (21)$$

where $\sigma_{share}=0.5$ is the size or radius of the niche and $\alpha=1$

$$nc_i = \sum_{j=1}^{\mu r_i} Sh(d_{ij}) \quad (22)$$

where μr_i is the number of solutions of rank r_i .

Step 4: Once the niche counts have been obtained, equation (23) is used to adjust the function, considering the fitness of each solution to obtain a shared fitness value, SF_i .

$$SF_i = \frac{F_i'}{nc_i} \quad (23)$$

Step 5: In order to maintain the average fitness of the solutions for each ranking before applying the shared fitness function, the values are scaled using equation (24) in order that their average value is the same as the average fitness value assigned.

$$SF_i' = \frac{F_i'}{\sum_{r_j=1}^{\mu(r_j)} SF_{r_j}} \mu(r_j) \quad (24)$$

4.1.4 Selection, Crossover and Mutation Parameters

Based on the fitness shown by each solution in the population, the Stochastic Universal Sampling (SUS) selection mechanism proposed by Baker (1987) is used. This mechanism uses a single random value to test all the solutions and choose them at uniformly spaced intervals. This allows

weaker members of the population the opportunity to be chosen and combined. For this algorithm, only $N/2$ solutions are selected, where N = number of solutions in the population.

The crossover process exchanges information on the different solutions with the intention of generating descendants that are better and of good quality. The crossover operation is carried out on one of the echelons of the supply chain and with an approval probability of 90%. Two previously selected solutions are considered, the first or second echelon of the chain is chosen at random to perform the crossover. If echelon 2 is chosen, the line of a plant is selected at random and swapped between the two solutions. If it is echelon 1, the line of a distribution centre is chosen at random and the same operation is performed, as shown in figure 3, adjusted to maintain solution feasibility. The process is repeated until the new population has been completed.

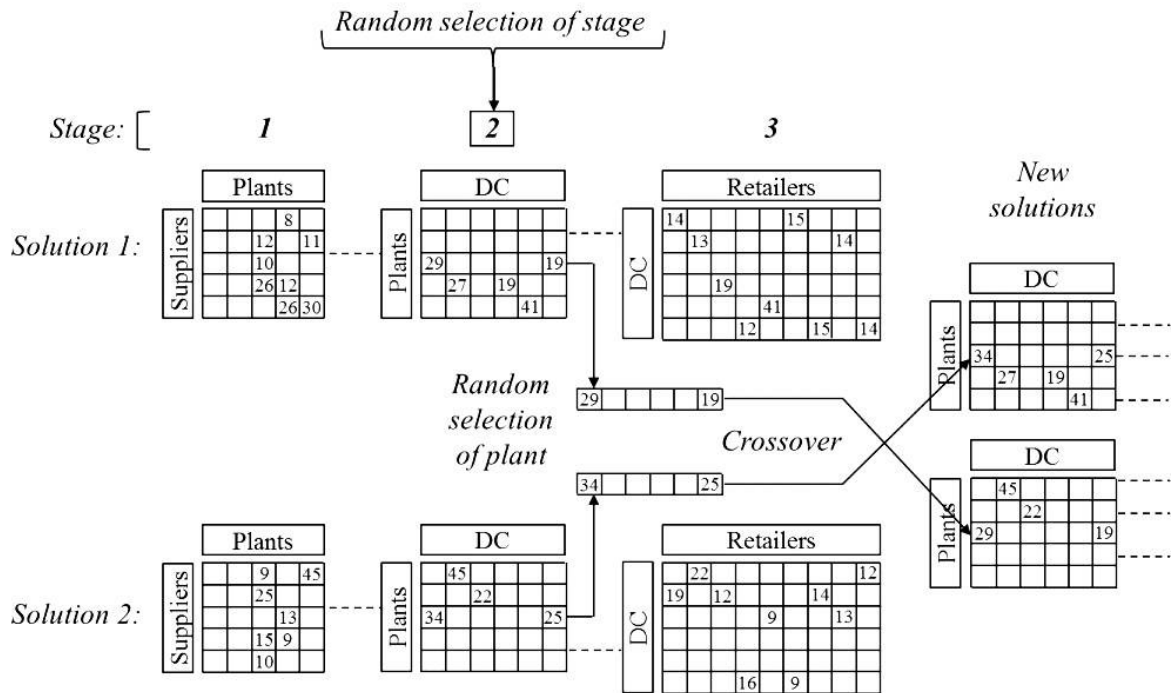


Fig. 3 Crossover.

The mutation process causes disturbance in the solution structure in order that the solutions avoid remaining at localised points. In this algorithm, the procedure is carried out on the first echelon of the chain, on the supplies sent from suppliers to plants, with an approval probability of 5%. One plant that receives goods from suppliers is chosen at random. The amount of material that a supplier sends to this plant is also selected at random, and this amount is reassigned to another supplier, considering solution feasibility, as shown in the illustration in figure 4.

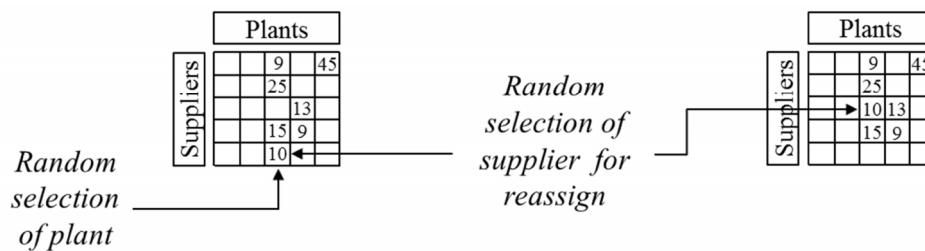


Fig. 4 Mutation.

4.1.5 Coding

As shown in figure 5, the first step is to generate a random initial population IP . The objective functions, EOF , are evaluated for each solution in the population. Each solution is assigned an initial fitness of ranking (r_j) according to the non-dominance criterion (1 is the best, 2 is the second best level, and so on). The fitness values of each solution are calculated in accordance with the shared fitness process. The best solutions with ranking 1 are selected. The first Pareto Front is generated. After the initial generation, the procedure continues from step 10 to form solution generations as shown in figure 5.

Multi-objective Evolutionary Algorithm

START

Input: Data from the SS-ILPSC model

Output: Set of non-dominated solutions.

1. Generate initial population (P_i)
 2. Evaluate objective functions (EOF)
 3. Assign Rank Based on Pareto Dominance (r_j)
 4. Shared fitness
 5. Calculate average fitness (F')
 6. Calculate niche count (NC)
 7. Calculate shared fitness (SF)
 8. Scale shared fitness (SCF)
 9. Select best solutions with ranking 1 (B_k)
 10. Generate Pareto Front
 11. For ($i=1$ up to G_n Generations)
 12. Select by Stochastic Universal Sampling (SUS)
 13. Crossover at a single point
 14. Mutation
 15. Evaluate objective functions of new population (P_{i+1})
 16. Assign Ranking Based on Pareto Dominance (r_j)
 17. Shared fitness
 18. Calculate average fitness (F')
 19. Calculate niche count (NC)
 20. Calculate shared fitness (SF)
 21. Scale shared fitness (SCF)
 22. Select best solutions with ranking 1 (B_{k+1})
 23. Unite best solutions of this generation with the previous ($B_k + B_{k+1}$)
 24. Assign Rank Based on Pareto Dominance (r_j)
 25. Select best solutions with ranking 1 (BoB_n)
 26. Generate Pareto Front
 27. End cycle
- END
-

Fig. 5. Multi-objective Evolutionary Algorithm

In order to form the Pareto fronts, before ending each generation, the best solutions with ranking 1 from the total population (B_k) were selected and unified with those of the previous generation to form a single set of solutions with ranking 1. Each solution in this set was immediately reassigned a new ranking in accordance with the non-dominance criterion. They were then filtered for the best solutions with ranking 1 (BoB_n) to form the approximate Pareto front for this generation and to be preserved. The process is repeated for each iteration until the algorithm ends. The Pareto front that is updated at each generation is the end product of the algorithm. A diagram of the procedure is shown in figure 6.

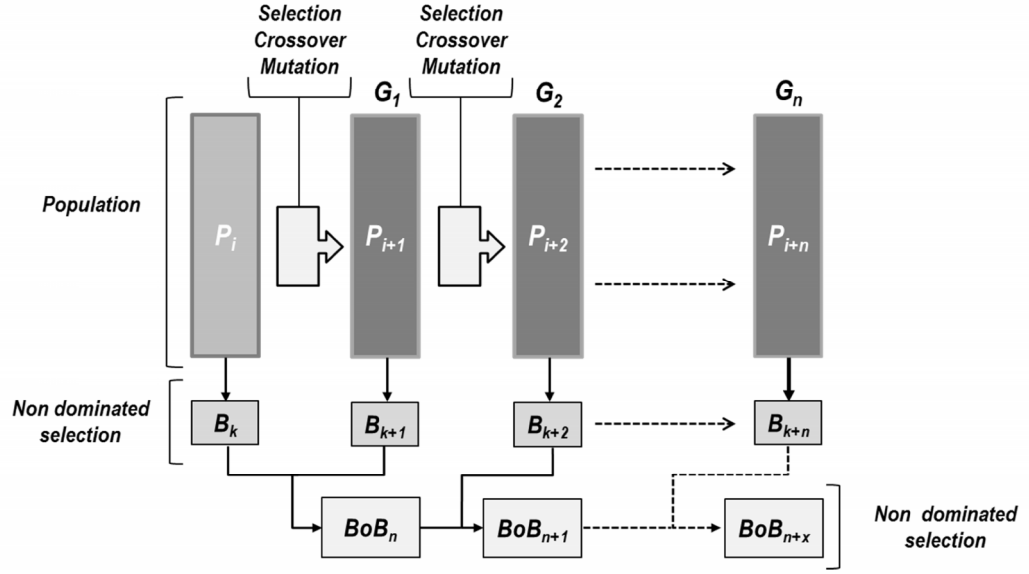


Fig. 6. Procedure of the Multi-objective Evolutionary Algorithm

4.2 Epsilon-constraint method using LINGO

The model was also solved using commercial optimisation software LINGO ver. 14.0, applying the epsilon-constraint technique which is one of the most typical methods for solving combinatorial multi-objective optimisation problems. This method generates non-dominated solutions to allow the person making the decision to choose one of them.

The epsilon-constraint model consists of the transformation of a model where one of the objective functions is already optimised and the others become part of the set of constraints (Ehrgott, 2005). In this case, the function, Z_2 , on supplier efficiency is incorporated into the set of constraints as shown in equation (25).

$$\sum_{i=1}^I \sum_{j=1}^J \frac{(\beta_{ij} \cdot OEE_i)}{\sum_{l=1}^L d_l} > \varepsilon \quad (25)$$

Where ε represents the average OEE percentage in the production volume with values between 0 and 1. The values for ε are systematically changed to obtain the Pareto front.

The LINGO optimisation software is then used to solve each sub-problem arising from the epsilon-constraint method. Due to the mixed-integer nonlinear nature of the model, it was decided to use this software as it obtains local or global optimal solutions for this type of problem. The default parameters set by the software were used. The technique applied by the software is the Generalised Reduced Gradient (GRG) with Successive Linear Programming (SLP)

5 Experimentation and Results

The following objectives were set when evaluating the effectiveness, efficiency and performance of the computation. The first is to value the use of frequently used commercial software and of a metaheuristic of an evolutionary algorithm with scenarios of different sizes, small and large, to observe the number of solutions in each Pareto front generated by each method, and in what time. The second goal it to present a comparison of the Pareto fronts using certain metrics.

In order to test the computational performance, the scenarios are generated at random as described by Guerrero (2013). The size of the scenarios is expressed using a group code as shown in column 1 of table 1, which is made up of four parameters: [the number of suppliers-the number of plants-the number of distribution centres-the number of customers].

5.1 Approximated solution sets using LINGO

The epsilon-constraint method was used to obtain reference Pareto fronts. The model was coded using the commercial optimisation software LINGO ver. 14.0. The ‘multistart solver’ option was used with a value of 80 attempts for each of the different values of ε , in each of the scenarios. For each value of ε , 10 runs were performed as the result varied between runs in terms of the objective function and the feasibility. The best solution was chosen from these 10 runs for each ε value. Of all the solutions found, only those that were feasible and non-dominated were considered. The time for all runs and the number of solutions found were recorded for comparison with the evolutionary algorithm (see tables 1 to 5).

In many cases only local optimums were obtained and occasionally no convergence was achieved in the maximum time of three hours per point using the epsilon-constraint method. For this reason, though it is expected that the solutions are of good quality, it can be said that each front is also an approximation of the true front for each scenario.

5.2 Approximate solution sets using the Evolutionary Algorithm

The results obtained using the evolutionary algorithm formed sets of non-dominated solutions approximate to the true Pareto front. All procedures were coded in Matlab version 7.12.0.632 and run on a PC model HP 6730s, with an Intel® Core™2 Duo, 2.00GHz and 2.0 GB of RAM. All the scenarios of the different groups were solved with 10 repetitions each. This is because the same result is not always obtained due to the random element of the evolutionary algorithm.

The following parameters were considered for each scenario: population size of 600 solutions, a total of 100 generations, combination percentage of 90% and mutation percentage of 5%. Ten repetitions were carried out for each of the different scenarios. The time for each repetition and the number of solutions found were recorded for comparison with the results from the commercial optimisation software LINGO ver. 14.0 (see tables 1 to 5). One characteristic of the fronts obtained is that they are discontinuous, due to the nature of the problem itself.

5.3 Comparisons

In order to be able to compare the fronts obtained with the MOEA and with LINGO, a metric is used. It consists of calculating the average Euclidian distance, DE_{Med} , between a reference point, P_r ,

(x_1, y_1) and different points, P_i , with coordinates (x_2, y_2) and which form the set of solutions on each of the fronts; it is obtained using equation (26).

$$DE_{Med}(P_r, P_i) = \frac{\sum_{i=1}^n (\sqrt{(x_i - x_1)^2 + (y_i - y_1)^2})}{n} \quad (26)$$

In this case, the reference point is (0,1), where the function Z_1 , corresponding to total costs, has a minimum value of 0 and the function Z_2 , corresponding to OEE, has a maximum value of 1. Figure 2 shows an example of how this procedure is carried out. Once the distance of each point is obtained the average of all the distances is calculated, thus giving an average value for each front, in each scenario, for comparison.

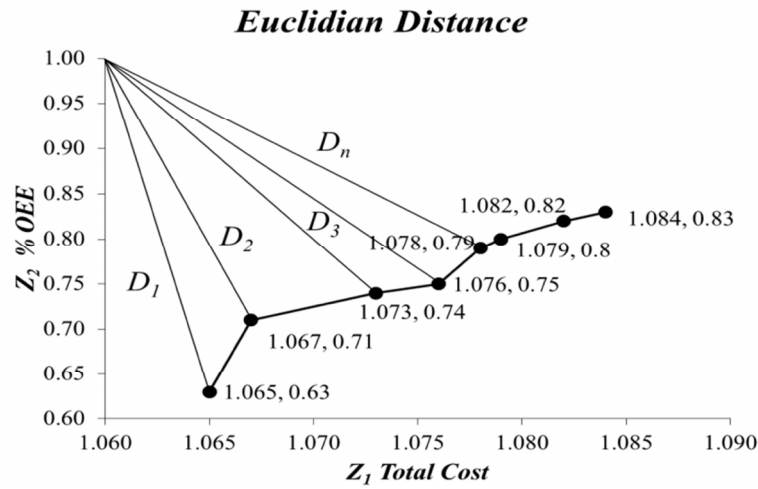


Fig. 7. Metric of Euclidian distance between a reference point and points on the front.

The *Rpos* percentage (Altıparmak et al. 2006) is also calculated. This is a metric used to compare two or more fronts and is calculated using equation (27). It involves building a front P by joining the solutions from the r sets obtained with different methods, omitting any dominated solutions. The result indicates the percentage of points on each front that belong to the set of solutions obtained with a particular method.

$$R_{pos}(S_i) = \frac{|\{s \in S_i \mid s \in P\}|}{|S_i|} \quad (27)$$

Tables 1 to 5 show the results of the comparison between the commercial software method and the evolutionary algorithm for the aforementioned scenarios. The first column of each table shows the group code of each scenario, and the following columns are identified with the two solution methods.

The results of the metaheuristic are identified as MOEA and those from the commercial software are identified as LINGO. For each method, the total run time, the number of feasible solutions and non-dominated solutions are shown along with the metrics of average distance and *Rpos*. The last row shows the general averages of each of the algorithms, which are used for the final comparison.

Scenario	MOEA				LINGO					
	Repetition	Total solutions	Time	D_{Med}	Total Repetitions	Total solutions	Total Time	D_{Med}	Rpos MOEA	Rpos LINGO
5-3-5-10	1	20	102.93	1.1197	10	2	219688.36	1.1929	1.00	0.00
	2	19	101.30	1.1190					1.00	0.00
	3	19	99.63	1.1167					1.00	0.00
	4	20	99.95	1.1183					1.00	0.00
	5	20	100.75	1.1193					1.00	0.00
	6	20	100.99	1.1228					1.00	0.00
	7	20	101.53	1.1197					1.00	0.00
	8	20	105.13	1.1211					1.00	0.00
	9	18	100.12	1.1163					1.00	0.00
	10	16	101.63	1.1135					1.00	0.00
				1.118	219688.3					
AV.		19.2	101.40	6		2	6	1.1929		

Table 1. Comparison of results with LINGO and the MOEA for scenario 5-3-5-10.

Scenario	MOEA				LINGO					
	Repetition	Total solutions	Time	D_{Med}	Total Repetitions	Total solutions	Total Time	D_{Med}	Rpos MOEA	Rpos LINGO
5-3-10-10	1	22	110.25	1.0827	10	4	121774.65	1.3015	1.00	0.00
	2	20	105.66	1.0774					1.00	0.00
	3	21	113.79	1.0764					1.00	0.00
	4	22	109.45	1.0871					1.00	0.00
	5	20	109.38	1.0729					1.00	0.00
	6	22	115.45	1.0912					1.00	0.00
	7	22	116.59	1.0809					1.00	0.00
	8	21	116.95	1.0902					1.00	0.00
	9	21	113.92	1.0759					1.00	0.00
	10	24	114.91	1.0877					1.00	0.00
AV.		21.5	112.64	1.0822	AV.	4	121774.7	1.3015		

Table 2. Comparison of results with LINGO and the MOEA for scenario 5-3-10-10.

Scenario	MOEA				LINGO					
	Repetition	Total solutions	Time	D_{Med}	Total Repetitions	Total solutions	Total Time	D_{Med}	Rpos MOEA	Rpos LINGO
5-3-10-15	1	9	102.01	1.3831	10	2	90314.64	1.6229	1.00	0.00
	2	9	100.14	1.3787					1.00	0.00
	3	9	99.08	1.3830					1.00	0.00
	4	9	100.18	1.3830					1.00	0.00
	5	9	97.96	1.3778					1.00	0.00
	6	10	103.09	1.3820					1.00	0.00
	7	9	104.26	1.3860					1.00	0.00
	8	9	102.81	1.3816					1.00	0.00
	9	9	103.88	1.3727					1.00	0.00
	10	8	102.52	1.3733					1.00	0.00
AV.		9	101.59	1.3801	AV.	2	90314.64	1.6229		

Table 3. Comparison of results with LINGO and the MOEA for scenario 5-3-10-15.

Scenario	MOEA				LINGO					
	Repetition	Total solutions	Time	D_{Med}	Total Repetitions	Total solutions	Total Time	D_{Med}	R_{pos} MOEA	R_{pos} LINGO
5-5-5-15	1	9	121.98	1.3883	10	4	42422.15	1.5226	1.00	0.00
	2	8	123.40	1.3877					1.00	0.00
	3	8	123.83	1.3867					1.00	0.00
	4	7	124.67	1.3880					1.00	0.00
	5	7	125.19	1.3869					1.00	0.00
	6	8	126.35	1.3880					1.00	0.00
	7	11	119.96	1.3893					1.00	0.00
	8	7	119.10	1.3878					1.00	0.00
	9	9	122.17	1.3880					1.00	0.00
	10	7	119.53	1.3863					1.00	0.00
AV.		8.1	122.62	1.3877	AV.	4	42422.15	1.5226		

Table 4. Comparison of results with LINGO and the MOEA for scenario 5-5-5-15.

Scenario	MOEA				LINGO					
	Repetition	Total solutions	Time	D_{Med}	Total Repetitions	Total solutions	Total Time	D_{Med}	R_{pos} MOEA	R_{pos} LINGO
5-3-10-15	1	4	135.47	1.9417	10	0*	2674.37			
	2	4	134.54	1.9420						
	3	5	143.87	1.9425						
	4	4	134.10	1.9642						
	5	4	137.23	1.9392						
	6	3	137.51	1.9413						
	7	4	138.84	1.9501						
	8	4	139.50	1.9525						
	9	5	137.37	1.9376						
	10	4	137.08	1.9571						
AV.		6.3	124.60	1.9878	AV.	0	2674.37			

*No feasible solutions found

Table 5. Comparison of results with LINGO and the MOEA for scenario 5-3-10-15.

The comparison of the results for each metric can be done in the following way: shorter times are better, a higher number of solutions is better, a lower average distance is better because it is closer to the ideal reference point and a higher R_{pos} is better. These values indicate the quality of the Pareto fronts. It can be seen that in 100% of cases, better results are obtained with the evolutionary algorithm than with LINGO software. The function to be optimised, Z_1 , contains quadratic sections and thus the results given by the optimisation software are local optimums. In addition, in many of the runs, feasible solutions were not found. Figure 7 shows a visual comparison for scenario 5-3-10-15.

For scenarios 5-3-5-10, 5-3-10-10, 5-3-10-15 and 5-5-5-15, the total number of repetitions or runs, and the execution times and average distances of each of the fronts obtained, are significantly better for the evolutionary algorithm than for the commercial software method. It can even be seen that in most cases R_{pos} (MOEA) is always higher than R_{pos} (LINGO).

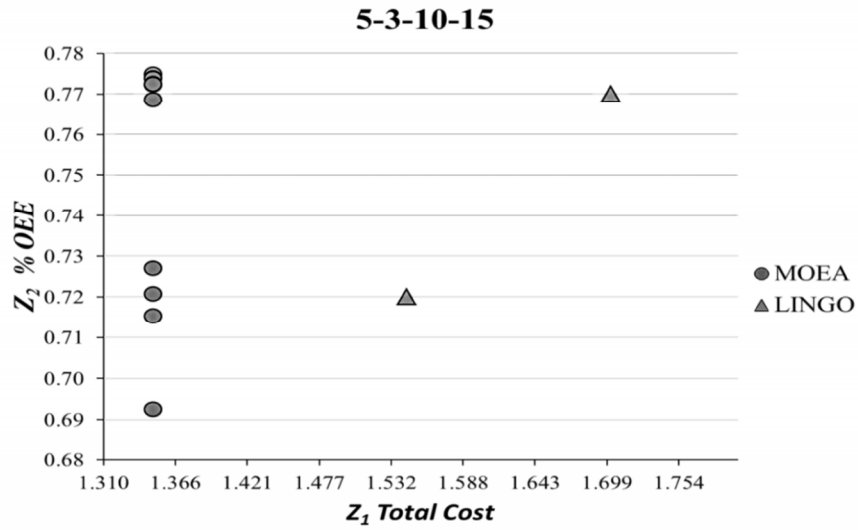


Fig. 7. Comparison of fronts for group scenario 5-3-10-15

In particular, for scenario 5-3-10-20, no front was obtained using the commercial software despite all the repetitions that were run, even over a considerably long time period. On the other hand, the evolutionary algorithm did find results in a reasonable run time, though it was not possible to compare these results with the other solution method to verify their quality.

6 Conclusions

Supply Chain Design involves elements related to decision-making regarding the location of facilities, the best number of facilities and the flow of products between them, as well as inventory control and product assignment to the different facilities, etc. As a result, it is important to obtain feasible and efficient results in a reasonable timeframe. These problems often have different and conflicting objectives, such as reducing costs, reducing time, increasing delivery effectiveness, improving level of service and increasing quality, among others.

This study presents results for a hypothetical though realistic supply chain design problem that aims to reduce costs on the entire chain and select the suppliers to provide materials to the different plants. The sets of solutions generated by means of an evolutionary algorithm are presented to the decision maker in terms of cost-efficiency trade-off.

The potential for solving large-scale and highly complex combinatorial problems using evolutionary algorithms is high, especially when dealing with a multi-objective situation. Four different small and medium-sized scenarios were tested to obtain sets of solutions in the form of Pareto fronts. One of the main advantages of the evolutionary algorithm is its relatively short run time and the number of solutions, in relation to the complexity and nature of the problem represented by a nonlinear model. It can be seen that for this type of nonlinear problems the solutions obtained with the commercial optimisation software LINGO ver. 14.0 are local optimums, and are not stable (repeatable and always feasible) or of good quality. The multi-objective evolutionary algorithm method is a good alternative for scenarios of any size because satisfactory results can be obtained in a reasonable time to enable decision-making.

For future studies related to the model, stochastic values can be introduced in plant capacity, distribution centre capacity, transportation costs and delivery times, etc., as these are important elements to consider when designing a supply chain, and also because real life scenarios are subject to constant change. The use of direct flows from plants to customers and between distribution centres may also be considered, or the case of multiple products and different transportation modes.

7 Acknowledgements

This paper is the result of doctoral studies conducted at *Universidad Popular Autonoma del Estado de Puebla* (UPAEP), with the support of a grant awarded by the National Council for Science and Technology (CONACYT).

References

1. Altiparmak, F., Gen, M., Lin, L., & Paksoy, T. (2006). A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers & Industrial Engineering* 51(1), 196-215.
2. Baker, James E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. *Proceedings of the Second International Conference on Genetic Algorithms and their Application*. Hillsdale, New Jersey, 14–21.
3. Coello Coello, C. A., Lamont, G., & Van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Nueva York: Springer.
4. Deb, K. (2008). *Multi-Objective Optimization using Evolutionary Algorithms*. United Kingdom: John Wiley & Sons, Ltd.
5. Fonseca, C., & Fleming, P. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Junio. San Francisco, CA, USA.
6. Forouzanfar, F., & Tavakkoli-Moghaddam, R. (2012). Using a genetic algorithm to optimize the total cost for a location-routing-inventory problem in a supply chain with risk pooling. *Journal of Applied Operational Research*, 4(1), 2-13.
7. Galbreth, M., Hill, J., & Handley, S. (2008). An investigation of the value of cross-docking for supply Chain Management. *Journal of Business logistics*, 29(1), 225-239.
8. Glover, F., & Kochenberger, G. (2003). *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers.
9. Guerrero C. A. (2013). *Supplier Selection in an Inventory Location Problem for a three-level Supply Chain*. Tesis Doctoral, UPAEP, Puebla, México.
10. Hnaien, F., Delorme, X., & Dolgui, A. (2010). Multi-objective optimization for inventory control in two-level assembly systems under uncertainty of lead times. *Computers & Operations Research*, 37(11), 1835-1843.
11. Javanshir, H., Ebrahimnejad, S., & Nouri, S. (2012). Bi-objective supply chain problem using MOPSO and NSGA-II. *International Journal of Engineering Computations*, 3(4), 681-694.
12. Ji, X., & Zhao, X. (2006). Supply Chain Network Design under Uncertainty. *Proceedings of the 11th Annual Conference of Asia Pacific Decision Sciences Institute*, 14-18 de junio. Hong Kong.
13. Karahan, I., & Köksalan, M. (2010). A Territory Defining Multiobjective Evolutionary Algorithms and Preference Incorporation. *IEEE Trans. Evolutionary Computation*, 14(4), 636-664.

14. Kim, J.-H., Han, J.-H., Kim, Y.-H., Choi, S.-H., & Kim, E.-S. (2012). Preference-Based Solution Selection Algorithm for Evolutionary Multiobjective Optimization. *IEEE Trans. Evolutionary Computation*, 16(1), 20-34.
15. Kleeman, M.P., Seibert, B.A., Lamont, G.B., Hopkinson, K.M., & Graham, S.R., (2007). Solving Multicommodity Capacitated Network Design Problems using a Multiobjective Evolutionary Algorithm. *IEEE Symposium on Computational Intelligence in Security and Defense Applications*, 1-5 de abril, Honolulu, Hawaii.
16. Konak, A., Coit, D., & Smith, A. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9), 992-1007.
17. Latha Shankar, B., Basavarajappa, S., C. H. Chen, J., & S. Kadadevaramath, R. (2013). Location and allocation decisions for multi-echelon supply chain network – A multi-objective evolutionary approach. *Expert Systems with Applications*, 40(2), 551-562.
18. Latha Shankar, B., Basavarajappa, S., S. Kadadevaramath, R., & C. H. Chen, J. (2013). A bi-objective optimization of supply chain design and distribution operations using non-dominated sorting algorithm: A case study. *Expert Systems with Applications*, 40(14), 5730-5739.
19. Li, L., & Schulze, L. (2011). Uncertainty in Logistics Network Design: A review. *Proceeding of the International Multiconference of Engineers and Computer Scientist*, 16-18 de marzo. Hong Kong.
20. Lim, M., & Xu, Y. (2005). Application of Evolutionary Algorithm in Supply Chain Management. *International Journal of Computers, Systems and Signals*, 6(1), 64-77.
21. Miandoabchi, E., Daneshzand, F., Szeto, W., & Zanjirani Farahani, R. (2013). Multi-objective discrete urban road network design. *Computers & Operations Research*, 40(10), 2429-2449.
22. Miranda, P., & Garrido, R. (2004). Incorporating inventory control decisions into a strategic distribution network design model with stochastic demand. *Transportation Research Part E*, 40(1), 183-207.
23. Miranda, P., Garrido, R., & Ceroni, J. (2009). e-Work based collaborative optimization approach for strategic logistic network design problem. *Computers & Industrial Engineering*, 57(1), 3-13.
24. Olivares Benitez, E., Ríos Mercado, R., & González Velarde, J. (2013). A metaheuristic algorithm to solve the selection of transportation channels in supply chain design. *International Journal of Production Economics*, 145(1), 161-172.
25. Purnomo, H.D., Wee, H.M., & Praharsi, Y. (2012). Two inventory review policies on supply chain configuration problem. *Computers & Industrial Engineering*, 63(2), 448-455.
26. Radhakrishnan, P., & Natarajan, J. (2011). Genetic Algorithm Model for Multi-factory Supply Chain Inventory Optimization involving Lead Time. *International Journal of Computational Engineering & Management*, 14(1), 142-150.
27. Radhakrishnan, P., Prasad, V., & Gopalan, M. (2009). Inventory Optimization in Supply Chain Management using Genetic Algorithm. *International Journal of computer Science and Network Security*, 9(1), 33-40.
28. Shu-Hsien Liao, Chia-Lin Hsieh, & Yu-Siang Lin. (2011). A multi-objective evolutionary optimization approach for an integrated location-inventory distribution network problem under vendor-managed inventory systems. *Annals of Operations Research*, 186(1), 213-229.
29. Sindhya, K., Miettinen, K., & Deb, K. (2013). A Hybrid Framework for Evolutionary Multi-Objective Optimization. *IEEE Trans. Evolutionary Computation*, 17(4), 495-511.
30. Stanley E., G., E. Bell, J., & J. Closs, D. (2012). Metaheuristics in Logistics and Supply Chain Management. *Journal of Business Logistics*, 33(2), 90-106.
31. Taskin Gumus, A., Fuat Guneri, A., & Keles, S. (2009). Supply chain network design using an integrated neuro-fuzzy and MILP approach: A comparative design study. *Expert Systems with Applications*, 36(10), 12570-12577.

32. Thu Bui, L., Michalewicz, Z., Parkinson, E., & Blanco Abello, M. (2012). Adaptation in Dynamic Environments: A Case Study in Mission Planning. *IEEE Trans. Evolutionary Computation* 16(2), 190-209.
33. Validi, S., Bhattacharya, A., & Byrne, P. (2014). A solution method for A two-layer sustainable supply chain distribution model. *Computers & Operations Research*, (In Press).
34. Venkatesan, S., Kumanan, S., & Karthik, S. (2009). Multi-criteria decision making in a supply chain using a genetic algorithm. *ICFAI Journal of Supply Chain Management*, 6(2), 67-82.
35. Wang, Y., & Cai, Z. (2012). Combining Multiobjective Optimization With Differential Evolution to Solve Constrained Optimization Problems. *IEEE Trans. Evolutionary Computation*, 16(1), 117-134.
36. Xing, L.-N., Rohlfshagen, P., Chen, Y.-W., & Yao, X. (2010). An Evolutionary Approach to the Multidepot Capacitated Arc Routing Problem. *IEEE Trans. Evolutionary Computation*, 14(3), 356-374.

5. CONCLUSIONES Y RECOMENDACIONES

En este trabajo se presentó el desarrollo de una metaheurística basada en un algoritmo evolutivo multiobjetivo para resolver el problema de diseño de una cadena de suministro con decisiones de localización-inventario y selección de proveedores representado en un modelo bi-objetivo de programación no lineal entero-mixto. Se muestran los resultados obtenidos para un hipotético pero realista problema de diseño de cadena de suministro. Se comparan estos resultados con los obtenidos en el software de optimización comercial LINGO ver. 14.0.

El potencial para resolver problemas combinatorios grandes mediante el uso de algoritmos evolutivos es alto, sobretodo cuando los casos son multiobjetivo. Cuatro diferentes instancias, pequeñas y medianas fueron probadas para obtener el conjunto de soluciones de los frentes de Pareto, en un tiempo razonable, considerando la complejidad y naturaleza del problema, demostrando que las metaheurísticas basadas en algoritmos evolutivos pueden ser utilizadas en este tipo de problemas.

Se observa que para este tipo de problemas no lineales, las soluciones obtenidas con el software de optimización comercial LINGO ver. 14.0, son óptimos locales y que no son estables (repetibles y siempre factibles) y de buena calidad. El método del algoritmo evolutivo multiobjetivo es una buena alternativa para instancias de cualquier tamaño.

La principal contribución de este trabajo es la manera de como se resuelve el problema presentado, utilizando algoritmos evolutivos para su solución, además de las adecuaciones que se hicieron al modelo original, donde ahora se consideran diferentes tiempos de entrega de cada proveedor a cada una de las diferentes plantas, que es una condición más realista. También se realizan adecuaciones al algoritmo empleado por Fonseca y Fleming (1993), como la de ir guardando y actualizando las mejores soluciones que se obtienen de generación en generación para crear el frente de Pareto aproximado. Se considera la métrica de la distancia euclidiana, que es un enfoque propuesto para comparar los diferentes frentes de Pareto obtenidos por el MOEA y el software comercial LINGO.

El MOEA aquí presentado, es susceptible de mejoras para que su desempeño sea aún más eficiente, considerando otros métodos de selección, combinación y mutación, así como realizar un estudio estadístico considerando diferentes valores en los parámetros empleados para su ejecución, como tamaño de la población, número de generaciones, porcentaje de combinación y mutación, que se puede realizar mediante un diseño de experimentos.

También, se pueden comparar los frentes usando métricas diferentes para evaluar si es posible definir la dominancia de un frente respecto a otro, obtenidos con diferentes métodos.

Para trabajos futuros relacionados con el modelo se pueden introducir valores estocásticos, en la capacidad de las plantas, capacidad de los centros de distribución, costos de transportación, tiempos de entrega, entre otros, que son importantes elementos a considerar al momento de diseñar una cadena de suministro, considerando que los escenarios de la vida real cambian constantemente. También puede ser considerado el uso de flujos directos de plantas a clientes así como entre centros de distribución, y considerar el caso de múltiples productos y diferentes medios de transporte.

Se recomienda utilizar métodos como el de la simulación para comprobar la robustez del modelo en la generación de los resultados, para tener mayor confianza al momento de tomar una decisión.

BIBLIOGRAFIA

- Altıparmak, F., Gen, M., Lin, L., & Paksoy, T. (2006). A genetic algorithm approach for multi-objective optimization of supply chain networks. *Computers & Industrial Engineering* 51(1), 196-215.
- Baker, James E. (1987). Reducing Bias and Inefficiency in the Selection Algorithm. *Proceedings of the Second International Conference on Genetic Algorithms and their Application*. Hillsdale, New Jersey, 14–21.
- Chopra, S., & Meindl, P. (2008). *Administración de la Cadena de Suministro*. Pearson, Prentice Hall.
- Coello Coello, C. A., Lamont, G., & Van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Nueva York: Springer.
- Deb, K. (2008). *Multi-Objective Optimization using Evolutionary Algorithms*. United Kingdom: John Wiley & Sons, Ltd.
- Ehrgott, M. (2005). *Multicriteria Optimization*. Heidelberg, Germany: Springer.
- Fonseca, C., & Fleming, P. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms*. Junio. San Francisco, CA, USA.
- Forouzanfar, F., & Tavakkoli-Moghaddam, R. (2012). Using a genetic algorithm to optimize the total cost for a location-routing-inventory problem in a supply chain with risk pooling. *Journal of Applied Operational Research*, 4(1), 2-13.
- Galbreth, M., Hill, J., & Handley, S. (2008). An investigation of the value of cross-docking for supply Chain Management. *Journal of Business logistics*, 29(1), 225-239.
- Ghiani, G., Laporte, G., & Musmanno, R. (2004). *Introduction to Logistics Systems Planning and Control*. England: John Wiley & Sons, Ltd.
- Glover, F., & Kochenberger, G. (2003). *Handbook of Metaheuristics*. Boston: Kluwer Academic Publishers.
- Guerrero C. A. (2013). *Supplier Selection in an Inventory Location Problem for a three-level Supply Chain*. Tesis Doctoral, UPAEP, Puebla, México.
- Hillier, F., & Hillier, M. (2009). *Introduction to management science a modeling and case studies approach with spreadsheets*. Boston: McGraw-Hill Co.
- Hnaien, F., Delorme, X., & Dolgui, A. (2010). Multi-objective optimization for inventory control in two-level assembly systems under uncertainty of lead times. *Computers & Operations Research*, 37(11), 1835-1843.

Javanshir, H., Ebrahimnejad, S., & Nouri, S. (2012). Bi-objective supply chain problem using MOPSO and NSGA-II. *International Journal of Engineering Computations*, 3(4), 681-694.

Ji, X., & Zhao, X. (2006). Supply Chain Network Design under Uncertainty. *Proceedings of the 11th Annual Conference of Asia Pacific Decision Sciences Institute*, 14-18 de junio. Hong Kong.

Karahan, I., & Köksalan, M. (2010). A Territory Defining Multiobjective Evolutionary Algorithms and Preference Incorporation. *IEEE Trans. Evolutionary Computation*, 14(4), 636-664.

Kim, J.-H., Han, J.-H., Kim, Y.-H., Choi, S.-H., & Kim, E.-S. (2012). Preference-Based Solution Selection Algorithm for Evolutionary Multiobjective Optimization. *IEEE Trans. Evolutionary Computation*, 16(1), 20-34.

Kleeman, M.P., Seibert, B.A., Lamont, G.B., Hopkinson, K.M., & Graham, S.R., (2007). Solving Multicommodity Capacitated Network Design Problems using a Multiobjective Evolutionary Algorithm. *IEEE Symposium on Computational Intelligence in Security and Defense Applications*, 1-5 de abril, Honolulu, Hawaii.

Konak, A., Coit, D., & Smith, A. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering & System Safety*, 91(9), 992-1007.

Latha Shankar, B., Basavarajappa, S., C. H. Chen, J., & S. Kadadevaramath, R. (2013). Location and allocation decisions for multi-echelon supply chain network – A multi-objective evolutionary approach. *Expert Systems with Applications*, 40(2), 551-562.

Latha Shankar, B., Basavarajappa, S., S. Kadadevaramath, R., & C. H. Chen, J. (2013). A bi-objective optimization of supply chain design and distribution operations using non-dominated sorting algorithm: A case study. *Expert Systems with Applications*, 40(14), 5730-5739.

Li, L., & Schulze, L. (2011). Uncertainty in Logistics Network Design: A review. *Proceeding of the International Multiconference of Engineers and Computer Scientist*, 16-18 de marzo. Hong Kong.

Lim, M., & Xu, Y. (2005). Application of Evolutionary Algorithm in Supply Chain Management. *International Journal of Computers, Systems and Signals*, 6(1), 64-77.

Lourenco, H. (2005). Logistics Management, an Opportunity for Metaheuristics. En C. Rego, & B. Alidaee, *Metaheuristics Optimization Via Memory and Evolution* (págs. 329-356). Boston: Kluwer Academic Publishers.

Miandoabchi, E., Daneshzand, F., Szeto, W., & Zanjirani Farahani, R. (2013). Multi-objective discrete urban road network design. *Computers & Operations Research*, 40(10), 2429-2449.

- Miranda, P., & Garrido, R. (2004). Incorporating inventory control decisions into a strategic distribution network design model with stochastic demand. *Transportation Research Part E*, 40(1), 183-207.
- Miranda, P., Garrido, R., & Ceroni, J. (2009). e-Work based collaborative optimization approach for strategic logistic network design problem. *Computers & Industrial Engineering*, 57(1), 3-13.
- Olivares Benitez, E., Ríos Mercado, R., & González Velarde, J. (2013). A metaheuristic algorithm to solve the selection of transportation channels in supply chain design. *International Journal of Production Economics*, 145(1), 161-172.
- Purnomo, H.D., Wee, H.M., & Praharsi, Y. (2012). Two inventory review policies on supply chain configuration problem. *Computers & Industrial Engineering*, 63(2), 448-455.
- Radhakrishnan, P., & Natarajan, J. (2011). Genetic Algorithm Model for Multi-factory Supply Chain Inventory Optimization involving Lead Time. *International Journal of Computational Engineering & Management*, 14(1), 142-150.
- Radhakrishnan, P., Prasad, V., & Gopalan, M. (2009). Inventory Optimization in Supply Chain Management using Genetic Algorithm. *International Journal of computer Science and Network Security*, 9(1), 33-40.
- Ragsdale, C. (2011). *Spreadsheet modeling and decision analysis 6th. ed.* Mason, OH: Cengage Learning
- Shu-Hsien Liao, Chia-Lin Hsieh, & Yu-Siang Lin. (2011). A multi-objective evolutionary optimization approach for an integrated location-inventory distribution network problem under vendor-managed inventory systems. *Annals of Operations Research*, 186(1), 213-229.
- Sindhya, K., Miettinen, K., & Deb, K. (2013). A Hybrid Framework for Evolutionary Multi-Objective Optimization. *IEEE Trans. Evolutionary Computation*, 17(4), 495-511.
- Stanley E., G., E. Bell, J., & J. Closs, D. (2012). Metaheuristics in Logistics and Supply Chain Management. *Journal of Business Logistics*, 33(2), 90-106.
- Taskin Gumus, A., Fuat Guneri, A., & Keles, S. (2009). Supply chain network design using an integrated neuro-fuzzy and MILP approach: A comparative design study. *Expert Systems with Applications*, 36(10), 12570-12577.
- Thu Bui, L., Michalewicz, Z., Parkinson, E., & Blanco Abello, M. (2012). Adaptation in Dynamic Environments: A Case Study in Mission Planning. *IEEE Trans. Evolutionary Computation* 16(2), 190-209.
- Validi, S., Bhattacharya, A., & Byrne, P. (2014). A solution method for A two-layer sustainable supply chain distribution model. *Computers & Operations Research*, (In Press).

Venkatesan, S., Kumanan, S., & Karthik, S. (2009). Multi-criteria decision making in a supply chain using a genetic algorithm. *ICFAI Journal of Supply Chain Management*, 6(2), 67-82.

Wang, Y., & Cai, Z. (2012). Combining Multiobjective Optimization With Differential Evolution to Solve Constrained Optimization Problems. *IEEE Trans. Evolutionary Computation*, 16(1), 117-134.

Xing, L.-N., Rohlfshagen, P., Chen, Y.-W., & Yao, X. (2010). An Evolutionary Approach to the Multidepot Capacitated Arc Routing Problem. *IEEE Trans. Evolutionary Computation*, 14(3), 356-374.

APENDICE A. Código en Matlab del Algoritmo Genético Multiobjetivo

Archivos:

MOGA_5_3_5_10.m	populinitial.m	e_PL_WH.m	cross.m
MOGA_5_3_10_10.m	WH_RT_2.m	e_SU_PL.m	edit_matrix.m
MOGA_5_3_10_15.m	PL_WH_2.m	e_SU_PL_z2.m	mutation.m
MOGA_5_5_5_15.m	SU_PL.m	ranking.m	selindbet.m
MOGA_5_5_10_20.m	evaluation.m	fitness.m	selnewind.m
intrnd.m	e_WH_RT.m	selection.m	

MOGA_5_3_5_10.m

%programa de un Multiobjective Genetic Algorithm

```
tic;
%supplier data
supplier = 5;
caps = [85    70    90    80    75];
UCs = [30    28    30    28    24];
OEEs = [0.85 0.5 0.9 0.7 0.6];

%warehouse data
warehouse = 5;
capw = [70    80    52    70    65];
FCw = [131250    150000 97500 131250 121875];
HCw = [184 220 190 189 205];
LTpw = [2 2 1 1 2; 1 2 1 2 1; 1 2 2 1 2];
OCpw = [200 200 200 200 200];

%plant data
plant = 3;
capp = [150 200 110];
FCp = [187500    250000 137500];
HCp = [173 130 236];
LTsp = [4 3 3; 2 2 2; 4 4 4; 2 2 3; 1 2 2];
OCsp = [500 500 500];

%retails data
retail = 10;
```

```

d = [30 25 26 25 20 28 27 17 27 25];
u = [106 95 195 168 135 123 106 117 108 99];

%transport costs
cost_PL_WH = [492 453 351 396 459;420 369 567 486 408;411 465 360 357 501];

cost_WH_RT= [282 297 258 225 216 240 270 225 222 270;210 258 249 270 225 240 222 225 282
240;234 258 294 231 231 213 294 261 237 252;300 270 240 234 249 297 282 216 240
225;279 267 291 273 261 249 297 294 285 297];

individual=600;
ratecross=0.9;
ratemut=0.05;
zt=1.96;

firstgen = populinitial (individual,supplier,plant,warehouse,retail,caps,capp,capw,d,u);
eval_pop = evaluation
(individual,supplier,plant,warehouse,retail,firstgen,FCp,HCp,LTsp,OCsp,FCw,HCw,LTpw,
OCpw,cost_PL_WH,cost_WH_RT,UCs,zt,OEEs,d);
populfitness = fitness (eval_pop,individual);
indbetter = selindbet (firstgen,populfitness,eval_pop,individual);
indbest = indbetter;
population = cat(2,firstgen,eval_pop,populfitness);
population (:,5) = [];population (:,7) = [];population (:,7) = [];
figure(1);

scatter(getcolumn(cell2mat(population(1:individual,5:6)),1),getcolumn(cell2mat(population
(1:individual,5:6)),2),'filled');
xlim([1.0 2]);ylim([0 1]);
title('Generacion: 0');

%comienza ciclo de generaciones

for g=1:100
    parents = selection (population,individual);
    newpopul = cross
        (individual,parents,population,supplier,plant,warehouse,retail,capp,caps,ratecross);

    [populmut,totalind] = mutation (newpopul,supplier, plant, ratemut);
    evalpopulmut = evaluation
        (totalind,supplier,plant,warehouse,retail,populmut,FCp,HCp,LTsp,OCsp,FCw,HCw,LTpw,
        OCpw,cost_PL_WH,cost_WH_RT,UCs,zt,OEEs,d);
    fitnesspopulmut = fitness (evalpopulmut,totalind);

    indbetter = selindbet (populmut,fitnesspopulmut,evalpopulmut,totalind);
    indbest = cat(1,indbest,indbetter);
    bestofbest = ranking (indbest);
    indbest = bestofbest;

```

```

figure(1);
scatter(getcolumn(cell2mat(bestofbest(:,5:6)),1),getcolumn(cell2mat(bestofbest(:,5:6)),2),'filled');
%xlim([1.0 1.2]);ylim([.5 .9]);
title(['Generacion: ',num2str(g)]);
grid on;

newpopulation = cat(2,populmut,evalpopulmut,fitnesspopulmut);
newpopulation(:,5) = [];newpopulation(:,7) = [];newpopulation(:,7) = [];
population = newpopulation;
end
toc;
disp(toc);

```

intrnd.m

```

function [ random ] = intrnd( level )

n = (1:level)';
element = [n (n-1)/level n/level];
rnd = rand;
random = find(element(:,2)<rnd & element(:,3)>rnd);

end

```

populinitial.m

```

function [ firstgen ] = populinitial(individual,supplier,plant,warehouse,retail,caps,capp,capw,d,u)

mat_SPWR = cell(individual,4);
for ind=1:individual
    mat_SPWR {ind,1} = ind;
    mat_WH_RT = WH_RT_2(warehouse,retail,capw,d,u);
    assig_WR = mat_WH_RT(1:warehouse,retail+1)';
    assig_v_WR = mat_WH_RT(1:warehouse,retail+3)';
    mat_SPWR {ind,4} = mat_WH_RT;

    mat_PL_WH = PL_WH_2(plant,warehouse,capp,assig_WR,assig_v_WR);
    assig_PW = mat_PL_WH(1:plant,warehouse+1)';
    assig_v_PW = mat_PL_WH(1:plant,warehouse+3)';
    max_assig_PW = max(assig_PW);
    mat_SPWR {ind,3} = mat_PL_WH;
    mat_SU_PL = SU_PL(supplier,plant,caps,assig_PW,max_assig_PW,assig_v_PW);
    mat_SPWR {ind,2} = mat_SU_PL;
end

firstgen = mat_SPWR;

end

```

WH_RT_2.m

```
function [ mat_WH_RT ] = WH_RT_2(warehouse,retail,capw,d,u)

%creación matriz almacenes - clientes
matrix_wr = zeros(warehouse+3,retail+3);
matrix_wr(1:warehouse,retail+2) = (capw)';
matrix_wr(warehouse+2,1:retail) = d;
matrix_wr(warehouse+3,1:retail) = u;

% inicia vectores de almacenes y clientes para asignaciones
totaldem = sum (d);
vectorWH=(1:warehouse);
lengthWH=numel(vectorWH);
newvectorWH = vectorWH;

vectorRT=(1:retail);
lengthRT=numel(vectorRT);
newvectorRT = vectorRT;

%ciclo de asignación demandas a almacenes
a=0;
b=0;
sumdemacum=0;
while sumdemacum < totaldem
    positionWH=intrnd(lengthWH-a);
    selectWH=newvectorWH(positionWH);
    full=0;
    while full == 0
        positionRT=intrnd(lengthRT-b);
        selectRT=newvectorRT(positionRT);
        otherRT = matrix_wr (warehouse+2,selectRT) + matrix_wr (selectWH,retail+1) ;
        if (otherRT <= matrix_wr (selectWH,retail+2))
            matrix_wr (selectWH,selectRT) = matrix_wr (warehouse+2,selectRT);
            newvectorRT (:,positionRT) = [];
            b=b+1;
            matrix_wr (selectWH,retail+1) = sum(matrix_wr(selectWH,1:retail),2);
            matrix_wr (warehouse+1,1:retail) = sum(matrix_wr(1:warehouse,1:retail),1);
        else
            pos=0;
            for w=1:retail
                if matrix_wr(warehouse+1,w) == 0
                    pos=pos+1;
                    otherRT2 = matrix_wr (warehouse+2,w) + matrix_wr (selectWH,retail+1);
                    if (otherRT2 <= matrix_wr (selectWH,retail+2))
                        matrix_wr (selectWH,w) = matrix_wr (warehouse+2,w);
                        newvectorRT (:,pos) = [];
                        b=b+1;
                        matrix_wr (selectWH,retail+1) = sum(matrix_wr(selectWH,1:retail),2);
                        matrix_wr (warehouse+1,1:retail) = sum(matrix_wr(1:warehouse,1:retail),1);
                        break;
                    end
                end
            end
        end
    end
    sumdemacum = sumdemacum + otherRT;
end
```

```

        end
    end
    end
    newvectorWH (:,positionWH) = [];
    a=a+1;
    full=1;
    end
    matrix_wr (selectWH,retail+1) = sum(matrix_wr(selectWH,1:retail),2);
    matrix_wr (warehouse+1,1:retail) = sum(matrix_wr(1:warehouse,1:retail),1);
    sumdemacum = sum(matrix_wr (warehouse+1,1:retail));
    end
    if isempty(newvectorWH)==1 && (sumdemacum < totaldem)
        matrix_wr (1:warehouse+1,1:retail+1) = 0;
        sumdemacum = sum(matrix_wr (warehouse+1,1:retail));
        a=0;
        b=0;
        newvectorWH = vectorWH;
        newvectorRT = vectorRT;
    end
end

%calculo de varianza de almacen
for k=1:warehouse
    vartot=0;
    for l=1:retail
        if matrix_wr (k,l) > 0
            vartot=vartot + matrix_wr(warehouse+3,l);
        end
    end
    matrix_wr (k,retail+3) = vartot;
end

mat_WH_RT = matrix_wr;

end

```

PL_WH_2.m

```
function [ mat_PL_WH ] = PL_WH_2(plant,warehouse,capp,assig_WR,assig_v_WR)
```

```

%creación matriz plantas - almacenes
matrix_pw = zeros(plant+3,warehouse+3);
matrix_pw(1:plant,warehouse+2) = (capp)';
matrix_pw(plant+2,1:warehouse) = assig_WR;
matrix_pw(plant+3,1:warehouse) = assig_v_WR;

```

```

% inicia vectores de almacenes y clientes para asignaciones
totaldem = sum (assig_WR);
vectorPL=(1:plant);
lengthPL=numel(vectorPL);

```

```

newvectorPL = vectorPL;

vectorWH=(1:warehouse);
h=1;
for wh=1:warehouse
    if matrix_pw (plant+2,wh)==0
        vectorWH (:,h) = [];
        h=h-1;
    end
    h=h+1;
end
lengthWH=numel(vectorWH);
newvectorWH = vectorWH;

%ciclo de asignación demandas a plantas
a=0;
b=0;
sumdemacum=0;
while sumdemacum < totaldem
    positionPL=intrnd(lengthPL-a);
    selectPL=newvectorPL(positionPL);
    full=0;
    while full == 0
        positionWH=intrnd(lengthWH-b);
        selectWH=newvectorWH(positionWH);
        otherWH = matrix_pw (plant+2,selectWH) + matrix_pw(selectPL,warehouse+1);
        if (otherWH <= matrix_pw (selectPL,warehouse+2))
            matrix_pw (selectPL,selectWH) = matrix_pw (plant+2,selectWH);
            newvectorWH (:,positionWH) = [];
            b=b+1;
            matrix_pw (selectPL,warehouse+1) = sum(matrix_pw(selectPL,1:warehouse),2);
            matrix_pw (plant+1,1:warehouse) = sum(matrix_pw(1:plant,1:warehouse),1);
        else
            pos=0;
            for p=1:warehouse
                if matrix_pw (plant+1,p) == 0 && matrix_pw (plant+2,p) > 0
                    pos=pos+1;
                    otherWH2 = matrix_pw (plant+2,p) + matrix_pw(selectPL,warehouse+1);
                    if (otherWH2 <= matrix_pw (selectPL,warehouse+2))
                        matrix_pw (selectPL,p) = matrix_pw(plant+2,p);
                        newvectorWH (:,pos) = [];
                        b=b+1;
                        matrix_pw (selectPL,warehouse+1) = sum(matrix_pw(selectPL,1:warehouse),2);
                        matrix_pw (plant+1,1:warehouse) = sum(matrix_pw(1:plant,1:warehouse),1);
                        break;
                    end
                end
            end
            newvectorPL (:,positionPL) = [];
            a=a+1;
            full=1;
        end
    end
end

```

```

    end
    matrix_pw (selectPL,warehouse+1) = sum(matrix_pw(selectPL,1:warehouse),2);
    matrix_pw (plant+1,1:warehouse) = sum(matrix_pw(1:plant,1:warehouse),1);
    sumdemacum = sum(matrix_pw (plant+1,1:warehouse));
end
if isempty(newvectorPL)==1 && (sumdemacum < totaldem)
    matrix_pw (1:plant+1,1:warehouse+1) = 0;
    sumdemacum = sum(matrix_pw (plant+1,1:warehouse));
    a=0;
    b=0;
    newvectorPL = vectorPL;
    newvectorWH = vectorWH;
end
end

%calculo de varianza planta
for j=1:plant
    vartot=0;
    for k=1:warehouse
        if matrix_pw (j,k) > 0
            vartot=vartot + matrix_pw(plant+3,k);
        end
    end
    matrix_pw (j,warehouse+3) = vartot;
end

mat_PL_WH = matrix_pw;

end

```

SU_PL.m

```

function [ mat_SU_PL ] = SU_PL(supplier,plant,caps,assign_PW,max_assign_PW,assign_v_PW)

%creación matriz plantas - almacenes
matrix_sp = zeros(supplier+3,plant+3);
matrix_sp(1:supplier,plant+2) = (caps)';
matrix_sp(supplier+2,1:plant) = assign_PW;
matrix_sp(supplier+3,1:plant) = assign_v_PW;

%ciclo de asignación demandas a almacenes
totaldem = sum (assign_PW);

vectorSP=(1:supplier);
lengthSP=numel(vectorSP);
newvectorSP = vectorSP;

vectorPL=(1:plant);
lengthPL=numel(vectorPL);
newvectorPL = vectorPL;

```

```

a=0;
b=0;
sumdemacum=0;
while sumdemacum < totaldem
    positionPL=intrnd(lengthPL-b);
    selectPL=newvectorPL(positionPL);
    full=0;
    while full == 0
        positionSP=intrnd(lengthSP-a);
        selectSP=newvectorSP(positionSP);
        difpl = matrix_sp(supplier+2,selectPL) - matrix_sp(supplier+1,selectPL);
        difsp = matrix_sp(selectSP,plant+2) - matrix_sp(selectSP,plant+1);
        if difpl <= difsp
            matrix_sp (selectSP,selectPL) = difpl;
            newvectorPL (:,positionPL) = [];
            b=b+1;
            full=1;
        else
            matrix_sp (selectSP,selectPL) = difsp;
            newvectorSP (:,positionSP) = [];
            a=a+1;
        end
        matrix_sp (selectSP,plant+1) = sum(matrix_sp(selectSP,1:plant),2);
        matrix_sp (supplier+1,1:plant) = sum(matrix_sp(1:supplier,1:plant),1);
        sumdemacum = sum(matrix_sp (supplier+1,1:plant));
    end
end

mat_SU_PL = matrix_sp;

end

```

evaluation.m

```

function [ eval_pop ] = evaluation
    (individual,supplier,plant,warehouse,retail,firstgen,FCp,HCp,LTsp,OCsp,FCw,HCw,LTpw,
    OCpw,cost_PL_WH,cost_WH_RT,UCs,z,OEEs,d)

eval_SPWR = cell(individual,3);
for i=1:individual
    eval_SPWR {i,1} = i;

    %evaluación Z1
    fl_WH_RT = firstgen {i,4};
    eval_WH_RT = e_WH_RT (warehouse,retail,cost_WH_RT,fl_WH_RT);
    fl_PL_WH = firstgen {i,3};
    eval_PL_WH =
        e_PL_WH(plant,warehouse,cost_PL_WH,fl_PL_WH,FCw,HCw,LTpw,OCpw,zt);

```

```

    fl_SU_PL = firstgen {i,2};
    eval_SU_PL = e_SU_PL (supplier,plant,fl_SU_PL,FCp,HCp,LTsp,OCsp,UCs,z);
    totalcost_Z1 = eval_WH_RT + eval_PL_WH + eval_SU_PL;
    eval_SPWR {i,2} = totalcost_Z1 ./ 1000000;

    %evaluación Z2
    eval_SU_PL_z2 = e_SU_PL_z2 (fl_SU_PL,supplier,plant,d,OEEs);
    eval_SPWR {i,3}= eval_SU_PL_z2;
end

eval_pop = eval_SPWR;

end

```

e_WH_RT.m

```

function [ eval_WH_RT ] = e_WH_RT (warehouse,retail,cost_WH_RT,fl_WH_RT)

%calcula de costo de almacenes a clientes
totalcost_WH_RT = 0;
for k=1:warehouse
    for l=1:retail
        totalcost_WH_RT = totalcost_WH_RT + (cost_WH_RT(k,l) * fl_WH_RT(k,l));
    end
end

eval_WH_RT = totalcost_WH_RT;

end

```

e_PL_WH.m

```

function [ eval_PL_WH ] = e_PL_WH
    (plant,warehouse,cost_PL_WH,fl_PL_WH,FCw,HCw,LTpw,OCpw,z)

%calcula de costos fijos de almacenes
tot_FCw=0;wicWH=0;ssWH=0;tot_costpw=0;
for k=1:warehouse
    if (fl_PL_WH(plant+1,k) > 0)
        %calcula de costos fijos
        tot_FCw = tot_FCw + FCw(k);
        %calcula de costo de inventario de trabajo en almacenes WIC
        wicWH = wicWH + (sqrt(2*OCpw(k)*HCw(k)) * sqrt(fl_PL_WH(plant+2,k)));
        for j=1:plant
            if fl_PL_WH(j,k) > 0
                %calcula de costo de stock de seguridad en almacenes SSC
                ssWH = ssWH + (zt * HCw(k) * sqrt(LTpw(j,k)) * sqrt(fl_PL_WH(plant+3,k)));
                %calcula de costos de flujo
                tot_costpw = tot_costpw + (fl_PL_WH(j,k) * cost_PL_WH(j,k));
            end
        end
    end
end

```

```

        end
    end
end
end

```

```
eval_PL_WH = tot_FCw + wicWH + ssWH + tot_costpw;
```

```
end
```

e_SU_PL.m

```
function [ eval_SU_PL ] = e_SU_PL (supplier,plant,fl_SU_PL,FCp,HCp,LTsp,OCsp,UCs,zt)
```

```
tot_FCp=0;wicPL=0;ssPL=0;tot_costsp=0;
```

```
vectorLTsp= zeros(supplier,plant);
```

```
maxLTsp= zeros(1,plant);
```

```
for jj=1:plant
```

```
    for ii=1:supplier
```

```
        if fl_SU_PL(ii,jj) > 0
```

```
            vectorLTsp(ii,jj) = LTsp(ii,jj);
```

```
        end
```

```
    end
```

```
    maxLTsp(jj) = max(vectorLTsp(:,jj));
```

```
end
```

```
%calcula el costo de suppliers a plants
```

```
for j=1:plant
```

```
    if (fl_SU_PL(supplier+1,j) > 0)
```

```
        %calcula de costos fijos
```

```
        tot_FCp = tot_FCp + FCp(j);
```

```
        %calcula de costo de inventario de trabajo en plantas WIC
```

```
        wicPL = wicPL + (sqrt(2*OCsp(j)*HCp(j)) * sqrt(fl_SU_PL(supplier+2,j)));
```

```
        %calcula de costo de stock de seguridad en plantas SSC
```

```
        ssPL = ssPL + (zt * HCp(j) * sqrt(maxLTsp(j)) * sqrt(fl_SU_PL(supplier+3,j)));
```

```
        for i=1:supplier
```

```
            if fl_SU_PL(i,j) > 0
```

```
                %calcula de costos de flujo
```

```
                tot_costsp = tot_costsp + (fl_SU_PL(i,j) * UCs(i));
```

```
            end
```

```
        end
```

```
    end
```

```
end
```

e_SU_PL_z2.m

```
function [ eval_SU_PL_z2 ] = e_SU_PL_z2 (fl_SU_PL,supplier,plant,d,OEEs)

totaldem = sum (d);
tot_OEEs=0;
for i=1:supplier
    tot_OEEs = tot_OEEs + ( (fl_SU_PL(i,plant+1) ./ totaldem) .* OEEs(i));
end

eval_SU_PL_z2 = tot_OEEs;

end
```

ranking.m

```
function [ paretoft ] = ranking( indbest )

nodomin = indbest;
[m,n]=size(nodomin);
tam=m;
for ind1=1:tam
    if ind1>tam; continue; end;
    for ind2=ind1+1:tam
        if ind2>tam; continue; end;
        if nodomin{ind1,5}==nodomin{ind2,5} && nodomin{ind1,6}==nodomin{ind2,6}
            nodomin(ind2,:)= [];
            tam=tam-1;
        end
    end
end
end
[mm,nn]=size(nodomin);

%CALCULAR RANKING DE CADA SOLUCION
rankfx = zeros(mm,2);
faux = zeros(mm,1);
for i=1:mm
    for j=1:mm
        if nodomin{j,5}<=nodomin{i,5}
            rankfx(i,1)=rankfx(i,1)+1;
            faux(j)=nodomin{j,6};
        else
            faux(j)=nodomin{i,6}-1;      %si es min-min se suma 1, si es min-max se resta 1
        end
    end
end
for jj=1:mm
    if (faux(jj)>=faux(i))      %si es min-min el signo es <=, si es min-max el signo es >=
        rankfx(i,2)=rankfx(i,2)+1;
    end
end
end
```

```

end

paretoft = nodomin;
paretoft(:,7) = num2cell(min(rankfx,[],2));

%elige solo los no dominados
i=1;
while i<=mm
    if cell2mat(paretoft(i,7)) ~= 1
        paretoft(i,:) = [];
        mm=mm-1;i=1;
    else
        i=i+1;
    end
end
end

end

```

fitness.m

```

function [populfitness] = fitness(eval_pop,individual)

populfitness = cell(individual,2);
fitnessproc = cell2mat(eval_pop);

%CALCULAR RANKING DE CADA SOLUCION
rankfx = zeros(individual,2);
faux = zeros(individual,1);
for i=1:individual
    for j=1:individual
        if (fitnessproc(j,2)<=fitnessproc(i,2))
            rankfx(i,1)=rankfx(i,1)+1;
            faux(j)=fitnessproc(j,3);
        else
            faux(j)=fitnessproc(i,3)-1;    %si es min-min se suma 1, si es min-max se resta 1
        end
    end
    for jj=1:individual
        if (faux(jj)>=faux(i))    %si es min-min el signo es <=, si es min-max el signo es >=
            rankfx(i,2)=rankfx(i,2)+1;
        end
    end
end
fitnessproc(:,4) = min(rankfx,[],2);

%CALCULAR EL PROMEDIO DE CADA SOLUCION
averfit = zeros(individual,5);
averfit(:,1) = fitnessproc(:,1);
averfit(:,2) = sort(fitnessproc(:,4),1);

```

```

averfit(:,3) = flipud(fitnessproc(:,1));
for i=1:individual
    for j=1:individual
        if (averfit(j,2) == averfit(i,1))
            averfit(i,4) = averfit(i,4) + 1;
            averfit(i,5) = averfit(i,5) + averfit(j,3);
        end
    end
    if (averfit(i,5) == 0)
        averfit(i,6) = 0;
    else
        averfit(i,6) = averfit(i,5) / averfit(i,4);
    end
end
for i=1:individual
    for j=1:individual
        if (fitnessproc(j,4) == averfit(i,1))
            fitnessproc(j,5) = averfit(i,6);
        end
    end
end
end

%DISTANCIA ENTRE LOS RANKINGS Y CONTADOR DE NICHOS
sigma=0.5;
alfa=1;
F1min=1;F1max=2;F2min=0;F2max=1;
niche = zeros(individual,5);
niche(:,1) = fitnessproc(:,1);
niche(:,2) = fitnessproc(:,2);
niche(:,3) = fitnessproc(:,3);
niche(:,4) = fitnessproc(:,4);
for i=1:individual
    %if niche(i,4) == 1
    for j=1:individual
        if (niche(j,4) == niche(i,4))
            distance = sqrt ( power (((niche(j,2) - niche(i,2)) / (F1max-F1min)),2) + power
            (((niche(j,3) - niche(i,3)) / (F2max-F2min)),2) );
            if (distance <= sigma)
                shared = 1-power((distance/sigma),alfa);
                niche(i,5) = niche(i,5) + shared;
            else
                niche(i,5) = niche(i,5) + 0;
            end
        end
    end
end
%else
%niche(i,5) = 1;
%end
end
fitnessproc(:,6) = niche(:,5);

```

```
%SHARED FITNESS
fitnessproc(:,7) = fitnessproc(:,5) ./ fitnessproc(:,6);
```

```
%SCALED FITNESS
scaled = zeros(individual,4);
scaled(:,1) = fitnessproc(:,1);
scaled(:,2) = fitnessproc(:,4);%ranking
scaled(:,3) = fitnessproc(:,5);%promedio
scaled(:,4) = fitnessproc(:,7);%shared fitness
factor = zeros(individual,3);
for i=1:individual
    factor(i,1) = i;
    miu=0;sh=0;F=0;
    for j=1:individual
        if (scaled(j,2) == i )
            F = scaled(j,3);
            miu = miu + 1;
            sh = sh + scaled(j,4);
        end
    end
    factor(i,2) = miu;
    factor(i,3) = (F*miu)/sh;
    if miu==0, factor(i,3) = 1; end
end
```

```
for i=1:individual
    for j=1:individual
        if factor(i,1) == scaled(j,2)
            fitnessproc(j,8) = fitnessproc(j,7) * factor(i,3);
        end
    end
    populfitness{i,1} = i;
    populfitness{i,2} = fitnessproc(i,4);
end
```

```
end
```

```
populfitness(:,3) = num2cell(fitnessproc(:,8));
```

```
end
```

selection.m

```
function [ parents ] = selection( population,individual )
%PROCESO DE SELECCION MEDIANTE EL METODO STOCHASTIC UNIVERSAL
SAMPLING SUS
```

```
% se genera la matriz sus con datos de xi, fi y scaled fitness
sus(:,1) = cell2mat(population(:,1));
sus(:,2) = cell2mat(population(:,7));
```

```

[tam,tcol]=size(population);

%se declaran valores de variables
npointer=(individual/2)+1;
distpointer= 1/npointer;
positionpointer = rand *(distpointer-0)+0;

%se calculan los acumulativos de scaled fitness y npointers
suma = sum(sus (:,2));
sus (:,3) = sus (:,2) / suma;
sus (:,4) = cumsum (sus(:,3),1);
sus (1,5) = positionpointer;
sus (2:tam,5) = distpointer;
sus (:,6) = cumsum (sus(:,5),1);

%se seleccionan los parents
parents = zeros(npointer,1);
solpar=1;
for i=1:npointer
    for j=1:tam
        if (j==1)
            if (sus(i,6)<=sus(j,4) && sus(i,6)>=0)
                parents (solpar,1) = sus(j,1);
                solpar=solpar+1;
            end
        else
            if (sus(i,6)<=sus(j,4) && sus(i,6)>=sus(j-1,4))
                parents (solpar,1) = sus(j,1);
                solpar=solpar+1;
            end
        end
    end
end
end
end
end

```

cross.m

```

function [ newpopul ] =
    cross(individual,parents,population,supplier,plant,warehouse,retail,capp,caps,ratecross )

newpopul = cell (individual,4);

pp=1;
for p=1:individual/2
    %elige la etapa de la cadena a cruzar
    selstage = fix(1+(3-1).*rand);
    parent1 = population {parents(p),selstage+1};
    parent2 = population {parents(p+1),selstage+1};

```

```

if rand < ratecross
    %identifica datos de la etapa elegida
    if selstage==1; selrow=supplier; col=plant; end
    if selstage==2; selrow=plant; col = warehouse; end
    if selstage==3; selrow=warehouse; col = retail; end
    pointcross = fix(1+(selrow-1).*rand);

    %genera los descendientes
    offspring1 = parent1;
    offspring2 = parent2;
    offspring1(pointcross,:) = parent2 (pointcross,:);
    offspring2(pointcross,:) = parent1 (pointcross,:);

    %modifica los descendientes para cumplir con demandas y capacidades
    newpopul {pp,1} = pp;
    newpopul {pp+1,1} = pp+1;
    if selstage == 3
        offspring1_mod = edit_matrix (offspring1,pointcross,selrow,col);
        newpopul {pp,4} = offspring1_mod;
        newassig_WR1 = offspring1_mod(1:warehouse,retail+1);
        newassig_v_WR1 = offspring1_mod(1:warehouse,retail+3);
        new_mat_PL_WH1 = PL_WH_2(plant,warehouse,capp,newassig_WR1,newassig_v_WR1);
        newpopul {pp,3} = new_mat_PL_WH1;
        newassig_PW1 = new_mat_PL_WH1(1:plant,warehouse+1);
        newassig_v_PW1 = new_mat_PL_WH1(1:plant,warehouse+3);
        max_newassig_PW1 = max(newassig_PW1);
        new_mat_SU_PL1 =
            SU_PL(supplier,plant,caps,newassig_PW1,max_newassig_PW1,newassig_v_PW1);
        newpopul {pp,2} = new_mat_SU_PL1;
        offspring2_mod = edit_matrix (offspring2,pointcross,selrow,col);
        newpopul {pp+1,4} = offspring2_mod;
        newassig_WR2 = offspring2_mod(1:warehouse,retail+1);
        newassig_v_WR2 = offspring2_mod(1:warehouse,retail+3);
        new_mat_PL_WH2 = PL_WH_2(plant,warehouse,capp,newassig_WR2,newassig_v_WR2);
        newpopul {pp+1,3} = new_mat_PL_WH2;
        newassig_PW2 = new_mat_PL_WH2(1:plant,warehouse+1);
        newassig_v_PW2 = new_mat_PL_WH2(1:plant,warehouse+3);
        max_newassig_PW2 = max(newassig_PW2);
        new_mat_SU_PL2 =
            SU_PL(supplier,plant,caps,newassig_PW2,max_newassig_PW2,newassig_v_PW2);
        newpopul {pp+1,2} = new_mat_SU_PL2;
    end

    if selstage == 2
        newpopul {pp,4} = population {parents(p),4};
        offspring1_mod = edit_matrix (offspring1,pointcross,selrow,col);
        newpopul {pp,3} = offspring1_mod;
        newassig_PW1 = offspring1_mod(1:plant,warehouse+1);
        newassig_v_PW1 = offspring1_mod(1:plant,warehouse+3);
        max_newassig_PW1 = max(newassig_PW1);

```

```

    new_mat_SU_PL1 =
    SU_PL(supplier,plant,caps,newassig_PW1,max_newassig_PW1,newassig_v_PW1);
    newpopul {pp,2} = new_mat_SU_PL1;
    newpopul {pp+1,4} = population {parents(p+1),4};
    offspring2_mod = edit_matrix (offspring2,pointcross,selrow,col);
    newpopul {pp+1,3} = offspring2_mod;
    newassig_PW2 = offspring2_mod(1:plant,warehouse+1)';
    newassig_v_PW2 = offspring2_mod(1:plant,warehouse+3);
    max_newassig_PW2 = max(newassig_PW2);
    new_mat_SU_PL2 =
    SU_PL(supplier,plant,caps,newassig_PW2,max_newassig_PW2,newassig_v_PW2);
    newpopul {pp+1,2} = new_mat_SU_PL2;
end

if selstage == 1
    newpopul {pp,4} = population {parents(p),4};
    newpopul {pp,3} = population {parents(p),3};
    newassig_PW1 = offspring1(supplier+2,1:plant)';
    newassig_v_PW1 = offspring1(supplier+3,1:plant);
    max_newassig_PW1 = max(newassig_PW1);
    offspring1_mod =
    SU_PL(supplier,plant,caps,newassig_PW1,max_newassig_PW1,newassig_v_PW1);
    newpopul {pp,2} = offspring1_mod;
    newpopul {pp+1,4} = population {parents(p+1),4};
    newpopul {pp+1,3} = population {parents(p+1),3};
    newassig_PW2 = offspring2(supplier+2,1:plant)';
    newassig_v_PW2 = offspring2(supplier+3,1:plant);
    max_newassig_PW2 = max(newassig_PW2);
    offspring2_mod =
    SU_PL(supplier,plant,caps,newassig_PW2,max_newassig_PW2,newassig_v_PW2);
    newpopul {pp+1,2} = offspring2_mod;
end
pp=pp+2;

else
    newpopul {pp,1} = pp;
    newpopul {pp+1,1} = pp+1;
    newpopul (pp,2:4) = population (parents(p),2:4);
    newpopul (pp+1,2:4) = population (parents(p+1),2:4);
    pp=pp+2;
end
end

end

```

edit_matrix.m

```
function [ offspring_mod ] = edit_matrix( offspring,pointcross,selrow,col )
```

```
%guarda el punto de cruza
```

```

savepointcross = offspring(pointcross,:);

%elimina doble asignación
offspring(1:selrow+1,1:col+1) = 0;
offspring(pointcross,:) = savepointcross;
offspring(1:selrow,col+1) = sum (offspring(1:selrow,1:col),2);
offspring(selrow+1,1:col) = sum (offspring(1:selrow,1:col),1);

    acum=0;
    for c=1:col
        if offspring(pointcross,c) > 0
            offspring(pointcross,c) = offspring(selrow+2,c);
            acum=acum+offspring(pointcross,c);
            if acum > offspring(pointcross,col+2)
                offspring(pointcross,c) = 0;
            end
        end
    end
end

%suma total de asignaciones
offspring(1:selrow,col+1) = sum (offspring(1:selrow,1:col),2);
offspring(selrow+1,1:col) = sum (offspring(1:selrow,1:col),1);

%asignación de demandas
totaldem = sum (offspring(selrow+2,1:col));
sumdemacum2=0;
nofac=0;
while sumdemacum2 < totaldem
    %inicialización de vectores
    vectorWH=(1:selrow);
    if nofac <= selrow*2
        vectorWH(:,pointcross)=[];
    end
    lengthWH=numel(vectorWH);
    newvectorWH = vectorWH;

    vectorRT=(1:col);
    e=1;
    for dest=1:col
        if offspring(pointcross,dest) > 0 && offspring(selrow+2,dest) > 0
            offspring(pointcross,dest) = offspring(selrow+2,dest);
            vectorRT(:,e) = [];
            e=e-1;
        end
        if offspring(selrow+2,dest) == 0
            offspring(pointcross,dest) = 0;
            vectorRT(:,e) = [];
            e=e-1;
        end
        e=e+1;
    end
end

```

```

lengthRT=numel(vectorRT);
newvectorRT = vectorRT;

%suma total de asignaciones
offspring(1:selrow,col+1) = sum (offspring(1:selrow,1:col),2);
offspring(selrow+1,1:col) = sum (offspring(1:selrow,1:col),1);

a=0;
b=0;
sumdemacum=0;sumdemacum2=0;
while sumdemacum < totaldem
    positionWH=intrnd(lengthWH-a);
    selectWH=newvectorWH(positionWH);
    full=0;
    while full == 0
        positionRT=intrnd(lengthRT-b);
        selectRT=newvectorRT(positionRT);
        otherRT = offspring (selrow+2,selectRT) + offspring(selectWH,col+1);
        if otherRT <= offspring (selectWH,col+2)
            offspring (selectWH,selectRT) = offspring (selrow+2,selectRT);
            newvectorRT (:,positionRT) = [];
            b=b+1;
            offspring (selectWH,col+1) = sum(offspring(selectWH,1:col),2);
            offspring (selrow+1,1:col) = sum(offspring(1:selrow,1:col),1);
        else
            pos=0;
            for p=1:col
                if offspring (selrow+1,p) == 0 && offspring (selrow+2,p) > 0
                    pos=pos+1;
                    otherRT2 = offspring (selrow+2,p) + offspring(selectWH,col+1);
                    if otherRT2 <= offspring (selectWH,col+2)
                        offspring (selectWH,p) = offspring (selrow+2,p);
                        newvectorRT (:,pos) = [];
                        b=b+1;
                        offspring (selectWH,col+1) = sum(offspring(selectWH,1:col),2);
                        offspring (selrow+1,1:col) = sum(offspring(1:selrow,1:col),1);
                        break;
                    end
                end
            end
            newvectorWH (:,positionWH) = [];
            a=a+1;
            full=1;
        end
        offspring (selectWH,col+1) = sum(offspring(selectWH,1:col),2);
        offspring (selrow+1,1:col) = sum(offspring(1:selrow,1:col),1);
        sumdemacum = sum(offspring (selrow+1,1:col));
        sumdemacum2 = sum(offspring (selrow+1,1:col));
    end
    if isempty(newvectorWH) && (sumdemacum < totaldem)
        nofac = nofac +1;
    end
end

```

```

        sumdemacum = totaldem;
        offspring(1:selrow+1,1:col+1) = 0;
        offspring(pointcross,:) = savepointcross;
    end
end
end

```

%calculo de la varianza

```

for k=1:selrow
    vartot=0;
    for l=1:col
        if offspring (k,l) > 0
            vartot=vartot + offspring (selrow+3,l);
        end
    end
    offspring (k,col+3) = vartot;
end

```

```

offspring_mod = offspring;

```

```

end

```

mutation.m

```

unction [ indbetter3,totalind ] = mutation( indbetter2,supplier, plant, ratemut )

```

```

totalind = numel(indbetter2(:,1));
indbetter3 = cell(totalind,4);
for i=1:totalind
    offspring = indbetter2{i,2};
    if rand < ratemut
        %elimina plantas vacias
        vectorPL=(1:plant);
        e=1;
        for jj=1:plant
            if offspring(supplier+2,jj) == 0
                vectorPL (,e) = [];
                e=e-1;
            end
            e=e+1;
        end
        lengthPL=numel(vectorPL);
        newvectorPL = vectorPL;
        positionPL=fix(1+(lengthPL-1).*rand);
        selplant=newvectorPL(positionPL);

        %elimina proveedores vacios
        vectorSP=(1:supplier);
        f=1;
        for ii=1:supplier

```

```

        if offspring(ii,selplant) == 0
            vectorSP(:,f) = [];
            f=f-1;
        end
        f=f+1;
    end
    lengthSP=numel(vectorSP);
    newvectorSP = vectorSP;
    positionSP=fix(1+(lengthSP-1).*rand);
    selsupp=newvectorSP(positionSP);

    units = offspring(selsupp,selplant);

    for sup=1:supplier
        if sup == selsupp || (offspring(sup,selplant) > 0);
            continue
        end
        newunits = units + offspring(sup,plant+1);
        if newunits <= offspring(sup,plant+2);
            offspring(sup,selplant) = units;
            offspring(selsupp,selplant) = 0;
            offspring (1:supplier,plant+1) = sum(offspring(1:supplier,1:plant),2);
            offspring (supplier+1,1:plant) = sum(offspring(1:supplier,1:plant),1);
            break
        end
    end
    end

    indbetter3{i,1} = i;
    indbetter3{i,2} = offspring;
    indbetter3(i,3) = indbetter2(i,3);
    indbetter3(i,4) = indbetter2(i,4);
else
    indbetter3{i,1} = i;
    indbetter3(i,2:4) = indbetter2(i,2:4);
end
end
end
end

```

selindbet.m

```

function [ indbetter ] = selindbet (firstgen,populfitness,eval_pop,individual)

indbetter = cell(individual,7);
cb=1;
for i=1:individual
    if cell2mat(populfitness(i,2)) == 1
        a = firstgen (i,:);
        b = eval_pop (i,2:3);
        c = populfitness (i,2);
    end
end

```

```

        indbetter (cb,1:7) = cat(2,a,b,c);
        cb = cb+1;
    end
end

```

```

indbetter (cb:individual,:) = [];

end

```

selnewind.m

```

function [ newpopulation ] = selnewind
    (individual,totalind,populmut,evalpopulmut,fitnesspopulmut)

newpopulation = cell(individual,7);

rank=1;
mem=1;
i=1;
while i<=individual
    if cell2mat(fitnesspopulmut(mem,2)) == rank
        newpopulation {i,1} = i;
        newpopulation (i,2:4) = populmut (mem,2:4);
        newpopulation (i,5:6) = evalpopulmut (mem,2:3);
        newpopulation (i,7) = fitnesspopulmut (mem,3);
        i=i+1;
    end
    mem=mem+1;
    if mem==totalind+1
        rank = rank+1;
        mem=1;
    end
end

end

end

```

APENDICE B. Código en LINGO ver.14.0

```

! -----
Notation and Definitions
Sets
i      :      Index for supplier, i=1,2,...,I.
j      :      Index for potential plant locations, j=1,2,...,J.
k      :      Index for potential warehouse locations, k=1,2,...,K.
l      :      Index for retailer, l=1,2,...,L.

Decision variables
W      :      1 if warehouse k serves retailer l, 0 otherwise.
X      :      1 if warehouse k is opened, 0 otherwise.
Y      :      1 if plant j is opened, 0 otherwise.
Z      :      1 if plant j serves warehouse k, 0 otherwise.

Auxiliary variables
BETA   :      Number of units of product produced for supplier i and
shipped to plant j
DELTA  :      Mean demand to be assigned to plant j
ETA    :      Variance of demand to be assigned to plant j
LAMDA  :      Mean demand to be assigned to warehouse k
PI     :      Variance of demand to be assigned to warehouse k

Parameters
cs     :      Production capacity of supplier i
cw     :      Maximum capacity of warehouse k
cp     :      Maximum capacity of plant j
d      :      Mean demand for each retailer l
FCp    :      Fixed cost of opening a plant at site j
FCw    :      Fixed cost of opening a warehouse at site k
HCp    :      Inventory-holding cost per unit in plant j
HCw    :      Inventory-holding cost per unit in warehouse k
LTp    :      Deterministic lead time for plant j orders
LTw    :      Deterministic lead time for warehouse k orders
OCp    :      Fixed ordering cost for a plant at site j
OCw    :      Fixed ordering cost for a warehouse at site k
OEEs   :      Overall Equipment Effectiveness for the supplier i
A      :      Transportation unit cost from plant j to warehouse k
B      :      Transportation unit cost warehouse k retailer l
UCs    :      Unit cost of producing product for supplier i
u      :      Mean variance for each retailer l
ZT     :      Value of standard normal distribution, which accumulates a
probability of 1-?
-----;

MODEL:
SETS:
SUPPLIER          : cs, OEEs, UCs;
PLANT             : cp, FCp, HCp, Y, DELTA, ETA, OCsp, LTSPMAX;
WAREHOUSE         : cw, FCw, HCw, X, LAMDA, PI, OCpw;
RETAILER          : d, u;
LINKSP(SUPPLIER,PLANT) : BETA, LTsp, R ;
LINKPW(PLANT,WAREHOUSE) : Z, A, LTpw;
LINKWR(WAREHOUSE,RETAILER) : W, B;
ENDSETS

```

```

@for(linksp(i,j):@if(beta(i,j)#GT#0,1,0) = R(i,j) );

!OBJECTIVE FUNCTION;

MIN =

@SUM(PLANT(J)      : FCp(J)*Y(J)
                        +
                        @SQRT(2*OCsp(J)*HCp(J))*@SQRT(DELTA(J)))
@SUM(LINKSP(I,J)   : ZT*HCp(J)*@SQRT(LTspMAX(J))*r(i,j)*@SQRT(ETA(J))
                        +UCs(I)* BETA(I,J))
                        +

@SUM(WAREHOUSE(K)  : FCw(K)*X(K)
                        +
                        @SQRT(2*OCpw(K)*HCw(K))*@SQRT(LAMDA(K)))
@SUM(LINKPW(J,K)   : ZT*HCw(K)*@SQRT(LTpW(J,K))*z(j,k)*@SQRT(PI(K))
                        +A(J,K)*z(j,k)*lamda(k))
                        +

@SUM(LINKWR(K,L)   : B(K,L)*d(L)*W(K,L));

!EPSILON CONSTRAINT;
@SUM(SUPPLIER(I): @SUM(PLANT(J): (BETA(I,J)/
@SUM(RETAILER(L):d(L))))*OEES(I)) >= E;
@SUM(SUPPLIER(I): @SUM(PLANT(J): (BETA(I,J)/
@SUM(RETAILER(L):d(L))))*OEES(I)) = Z2;
Z2 >= E;

!RESTRICTIONS;
@FOR(RETAILER(L)   : @SUM(WAREHOUSE(K) : W(K,L)) =
1);
@FOR(WAREHOUSE(K) : @SUM(PLANT(J)      : Z(J,K)) =
X(K));
@FOR(WAREHOUSE(K) : @SUM(RETAILER(L)   : d(L) * W(K,L)) =
LAMDA(K));
@FOR(WAREHOUSE(K) : @SUM(RETAILER(L)   : u(L) * W(K,L)) =
PI(K));
@FOR(PLANT(J)      : @SUM(WAREHOUSE(K) : LAMDA(K)*Z(J,K)) =
DELTA(J));
@FOR(PLANT(J)      : @SUM(WAREHOUSE(K) : PI(K) *Z(J,K)) =
ETA(J));
@FOR(PLANT(J)      : @SUM(SUPPLIER(I)   : BETA(I,J)) =
DELTA(J));
@FOR(WAREHOUSE(K)  : LAMDA(K) <=
cw(K) * X(K));
@FOR(PLANT(J)      : DELTA(J) <=
cp(J) * Y(J));
@FOR(PLANT(J)      : @SUM(SUPPLIER(I)   : BETA(I,J)) <=
cp(J) * Y(J));
@FOR(SUPPLIER(I)   : @SUM(PLANT(J)      : BETA(I,J)) <=
cs(I)) ;
@FOR(WAREHOUSE(K)  : @BIN(X));
@FOR(PLANT(J)      : @BIN(Y));
@FOR(LINKWR(K,L)   : @BIN(W));
@FOR(LINKPW(J,K)   : @BIN(Z));
@FOR(PLANT(J):@FOR(SUPPLIER(I): LTspmax(J) >= LTsp(I,J)*R(I,J));
@FOR(LINKSP(I,J)   : @GIN(BETA));

```

DATA:

E = 0.80 ; !E= 0.0 - 1.0;
ZT = 1.96; !ZT= 97.5% - 1.96, 90.0% - 1.28, 75 % - 0.67, 50 % - 0;

SUPPLIER = S1..S5;
PLANT = P1..P3;
WAREHOUSE = WH1..WH5;
RETAILER = R1..R10;

cs = 85 70 90 80 75 ;
UCs = 30 28 30 28 24 ;
OEEs = 0.85 0.5 0.9 0.7 0.6 ;

cp = 150 200 110 ;
FCp = 187500 250000 137500 ;

HCp = 173 130 236 ;

LTsp =
4 3 3
2 2 2
4 4 4
2 2 3
1 2 2 ;

OCsp = 500 500 500 ;

cw = 70 80 52 70 65 ;
FCw = 131250 150000 97500 131250 121875 ;

HCw = 184 220 190 189 205 ;

LTpw =
2 2 1 1 2
1 2 1 2 1
1 2 2 1 2 ;

OCpw = 200 200 200 200 200 ;

d = 30 25 26 25 20 28 27 17 27 25 ;
u = 106 95 195 168 135 123 106 117 108 99 ;

A =
492 453 351 396 459
420 369 567 486 408
411 465 360 357 501 ;

B =
282 297 258 225 216 240 270 225 222 270
210 258 249 270 225 240 222 225 282 240
234 258 294 231 231 213 294 261 237 252
300 270 240 234 249 297 282 216 240 225
279 267 291 273 261 249 297 294 285 297 ;

ENDDATA

END

APENDICE C. Certificado CLAIO 2014



**CLAIO
Monterrey 2014**

**XVII Latin-Iberian-American
Conference on Operations Research**

This is to certify that

Rodolfo Eleazar Pérez Loaiza

has presented the paper (ID 025) entitled:

“Diseño de una cadena de suministro con decisiones de eficiencia, localización e inventario usando un algoritmo evolutivo multiobjetivo”

in the XVII Latin-Iberian-American Conference on Operations Research (CLAIO), held in Monterrey, Mexico, October 6-10, 2014.

Roger Z. Ríos
Chair CLAIO XVII

José Luis González Velarde
Chair CLAIO XVII



**APENDICE D. Carta de recepción del editor del artículo en
Journal**