UNIVERSIDAD NACIONAL DE COLOMBIA

Optimización multi-objetivo con algoritmos bio-inspirados para el control y coordinación de inventarios multi-producto

Daniel Stevenson Grass Guaqueta

Universidad Nacional de Colombia
Facultad De Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2016

Optimización multi-objetivo con algoritmos bio-inspirados para el control y coordinación de inventarios multi-producto

Daniel Stevenson Grass Guaqueta

Tesis o trabajo de grado presentado como requisito parcial para optar al título de:
Magister en Ingeniería de Sistemas y Computación

Director:
Ph.D. Carlos Eduardo Alonso Malaver

Línea de Investigación:
Sistemas y organizaciones
Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2016

(Dedicatoria o un lema)

A mi Mami por su cuidado y amor. A mi hermano Jeser por sus consejos y horas de juego, a mi hermana Karen por sus sabias palabras y su ternura. y mi Papá Henry por todo lo que me has dado, sin ustedes esto no sería posible.

A la mujer que transformo mi vida, en ilusiones y pasión. A Mi amada esposa Paola por su apoyo y amor incondicional en todos los aspectos de mi vida.

A un pequeño gigante, Mi hijo Alexander por alegrarme mi vida.

Agradecimientos

En primer lugar agradezco al Profesor Carlos Eduardo Alonso Malaver, por su compromiso y enseñanza en todo este proceso.

A mi amigo Daniel Santiago por que juntos nos conocimos y nos apoyamos en esta travesía.

A la Universidad Nacional de Colombia por brindarnos el conocimiento, los docentes y el espacio para el crecimiento personal y académico.

Resumen

En las organizaciones es de vital importancia el control y manejo de los inventarios, dado que generan costos en su gestión y a su vez se relacionan con los niveles de servicio (satisfacción de sus clientes). En este trabajo se propone un entorno de decisión múlti-criterio. Se estudia y resuelve un problema de optimización multi-objetivo para la coordinación en la entrega de múltiples productos con demandas y tiempos de entrega bajo incertidumbre, minimizando los costos de la gestión y a su vez maximizando los niveles de servicio.

El problema propuesto se soluciona mediante algoritmos bio-inspirados. En primer lugar se analiza el comportamiento de las funciones objetivo de manera separada, utilizando Algoritmos Genéticos (AG), Optimización por enjambre de particulas (PSO), Cuckoo Search (CS) y Optimización por forrajeo de bacterias (BFOA). En segundo lugar se aplicaron los algoritmos multi-objetivo NSGA II, NSPSO, MOCS y BCMOA obteniendo la frontera óptima de Pareto. Las soluciones encontradas se comparan mediante las métricas de indicador de dominancia de Pareto, diversidad y tiempo computacional. Finalmente se aplicó TOPSIS seleccionando el punto más cercano a la solución ideal. En este sentido el entorno de decisión múlti-criterio consiste de la formación de la frontera de Pareto resultante y de la selección de una alternativa mediante TOPSIS.

Palabras clave: Optimización Multi-Objetivo, Gestión de inventarios, Algoritmos bio-inspirados, Coordinación multiproducto.

Abstract

Inventory management has achieved a prominent place in bussiness administration, relevance that is attached to costs reduction and maximization of customer satisfaction. In this work is proposed a framework multicriteria decision. Solving a problem of multi-objetive optimization inventory management and coordination of multiple products delivery.

The problem is solved using bio-inspired algorithms. Initially, we study the behavior of the uni-objective function case, using genetic algorithms (GA), particle swarm optimization (PSO), Cuckoo Search (CS) and Bacterial Foraging Optimization Algorithm (BFOA), and comparing their performance. After that we study multiobjective algorithms. We apply NSGA II, NSPSO, MOCS and BCMOA. The solutions are compared by the metric indicator Pareto dominance , diversity and computing time. Finally we apply TOPSIS algorithm for selecting the best solution.

Keywords: Multi-objetive Optimization, Inventory Management, Bio-inspired Algorithms, Coordination Multiple Products delivery.

Contenido

Lista de símbolos

Símbolos

Símbolos	Término
Ψ	Minimizar
ζ	Diferentes funciones objetivo
$\oplus$	Suma por cada dimensión
δ	Función de costo de inventario y pedido coordinado
ϑ	Función de frecuencia esperada de ocasiones de desabastecimiento anuales
φ	Función de la cantidad esperada de productos faltantes anualmente
ν	Función de costo de transporte del inventario
$\Im$	Frontera de Pareto

Abreviaturas

Abreviatura	Término
AG	Algoritmos genéticos
PSO	Optimización por enjambre de partículas
CS	Cuckoo Search
$BFOA$	Algoritmo de optimización por Forrajeo De Bacterias
$NSGA$	Elitist Non-dominated Sorting Genetic Algorithm
$NSPSO$	Non-dominated Sorting Particle Swarm Optimizer
$MOCS$	Multi-Objetive Cuckoo Search
$BCMOA$	Bacterial Chemotaxis Multiobjetive Optimization Algorithm
$TOPSIS$	Técnica para el ordenamiento preferencial por similitud con la solución ideal
JRP	Joint replenishment problem

1. Introducción

Comúnmente los inventarios están relacionados con la manutención de cantidades suficientes de recursos dentro de una organización para desarrollar un proceso o prestar un servicio, es decir, las organizaciones tienen entradas y salidas de bienes. Cuando estos flujos son almacenados - retenidos por un período de tiempo en algún lugar, se les denomina inventarios en tanto se consideran útiles para la organización aunque se encuentran en estado ocioso en un momento determinado [45].

1.1. Antecedentes y justificación

Este trabajo se dedica al estudio de formas eficientes de gestionar un inventario mediante el uso de algoritmos desarrollados a partir de comportamientos observados en la naturaleza (bio-inspirados).

El propósito al desarrollar un sistema de gestión de inventarios es plantear las políticas o reglas que se deben seguir para satisfacer la demanda del cliente y al tiempo tener períodos de ocio tan cortos como sea posible. Por tal motivo, el objetivo fundamental en estos problemas, es resolver las preguntas: ¿Con qué frecuencia debe revisarse el nivel de inventario?, ¿Cuándo debe ordenar, y qué cantidad debe ordenarse en cada pedido? [52].

Para resolver, estos interrogantes, es necesario conocer los principales componentes de un problema de inventario, como son: **Demanda**, la cual se define como el número de unidades requeridas en un periodo de tiempo determinado; **Productos**, que implica conocer el portafolio ofrecido por la organización, y, El **Tiempo de entrega**, i.e. el lapso de tiempo que transcurre entre la orden de compra o producción y el momento en el que se entrega o se termina de fabricar. La interpretación y relación adecuada de estos elementos permite entender el comportamiento dinámico del sistema y generar estrategias para el control adecuado de bienes o servicios producidos, almacenados y distribuidos.

Con una alta frecuencia en los problemas de inventarios, sólo se tiene en cuenta como función a minimizar el costo total anual de inventario (anual es sólo un periodo de referencia puede darse mes o semana), pero se ha observado desde la práctica que los sistemas no tienen una sola función a optimizar sino varias; y éstas pueden llevar a distintos óptimos que

en el momento de ser aplicados pueden llevar a decisiones que entran en conflicto entre sí. Verbigracia, en un sistema de control de inventario se debe operar a menor costo y mantener el nivel de servicio deseable para los clientes. Los objetivos de minimización de costos y la maximización del nivel de servicio son inconmensurables y entran en conflicto entre sí [56]. Este tipo de problemas es conocido en la literatura como un problema de optimización multi-objetivo (MOOP, multi-objective optimization problem) al caracterizarse por tener dos o más funciones a optimizar.

Los algoritmos inicialmente propuestos para resolver un MOOP, se basaban en métodos de agregación, en los cuales se solucionan por separado problemas mono-objetivo, y posteriormente se busca una solución óptima multi-objetivo a partir la combinación lineal de los óptimos mono-objetivo obtenidos previamente, o buscan optimizar un solo objetivo y los demás se traducen en restricciones [56]. Por lo tanto, los tomadores de decisiones enfrentan desafíos no sólo modelando el problema en un contexto multi-objetivo, sino también en el esfuerzo dedicado a construir el conjunto de soluciones óptimas de Pareto (conjunto de decisiones no dominadas).

En busca de solucionar el problema anterior se ha implementado un método de clasificación llamada Técnica para la orden de preferencia por similitud con solución ideal (TOPSIS-Technique for Order Preferences by Similarity to an Ideal Solution) [26, 32, 58] que permite clasificar las soluciones no dominadas de acuerdo al conocimiento-imposición del mercado o a la preferencia de los tomadores de decisiones.

Algunos de los trabajos recientes en modelos de inventario multi-objetivo con productos que se deterioran en el tiempo son: Padmanabhan y Vrat [40] quienes resuelven éste problema utilizando un método de programación no lineal por metas. Agrell [2] presentó un sistema de ayuda a la decisión multi-criterio para el control de inventario, en el que se presenta un procedimiento interactivo donde las preferencias son extraídas progresivamente en el proceso de análisis de decisiones para determinar el tamaño del lote y el inventario de seguridad.

Bookbinder y Chen [27] analizan sistemas de inventario multi-nivel usando MCDM ("Multi criterion decision making"), en Puerto y Fernández [46] obtienen un conjunto de Pareto de soluciones óptimas para problemas de inventario estocásticos, en Roy y Maiti [48] se formula un modelo de inventario multi-objetivo con productos que se deterioran con acciones dependientes a la demanda en virtud de área de almacenamiento la cual es imprecisa y limitada al costo total del presupuesto. Los objetivos en el mismo, son de maximizar el beneficio y reducir al mínimo el costo de desperdicio. Donde el objetivo de beneficio, costo de desperdicio y el área de almacenamiento son difusos en la naturaleza. El problema fue resuelto utilizando "Fuzzy no Lineal Programming "(FNLP) y "Fuzzy Aditive Goal Programming "(FAGP).

En Mahapatra y Maiti [35] consideran un modelo multi-objetivo de inventario con artículos que se deterioran estocásticamente e incorporan el impacto de nivel de calidad en la demanda y una función de deterioro. En Mandal et. al [36] presentaron un modelo de inventario difuso multi-producto con múltiples-objetivos. El modelo tiene tres restricciones para encontrar la demanda, el tamaño de la orden, y el nivel de escasez para cada elemento. Problema que es resuelto utilizando el método de programación geométrica.

Como se mencionó anteriormente, los estudios anteriores usan un vector de peso relativo para escalar los múltiples objetivos en un solo objetivo y solucionar el problema correspondiente mediante una sola técnica de optimización mono-objetivo.

Trabajos recientes en optimización multi-objetivo en problemas de inventario son de Tsou [56, 57], Pasandideh [42], Park y Kyung [41]. En Tsou [56] por primera vez se realiza optimización por enjambre de partículas multi-objetivo (MOPSO), algoritmo para generar el conjunto de soluciones no dominadas de un sistema de cantidad de pedido y orden; aplicando TOPSIS para generar la solución ideal para la toma de decisiones. En este trabajo se realiza la minimización del costo anual total esperado, la minimización de la frecuencia esperada anual de ocasiones de desabastecimiento, y la minimización de la cantidad esperada anual de desabastecimiento.

Tsou [57] usa un algoritmo mejorado de enjambre de partículas multi-objetivo que determina las fronteras de Pareto para sistemas de inventario de revisión continua. Pasandideh [42] desarrolla un modelo bi-objetivo del modelo de cantidad económica de producción para productos defectuosos usando el algoritmo de "non dominated sorting genetic algorithm (NSGA II)"y MOPSO. Park y Kyung[41] desarrollan un modelo de inventario multi-objetivo para obtener costos óptimos y una tasa de cumplimiento fijada de antemano usando MOPSO. Recientemente en [54] utilizan Cuckoo Search [60, 61] como un algoritmo para resolver problemas de inventario multi-objetivo, mostrando un buen desempeño.

Coello - Coello y Lechuga [10, 8] plantean y estudian el MOPSO, y concluyen que este algoritmo es una alternativa viable para resolver un MOOP, y a la vez los resultados obtenidos son similares a los obtenidos con NSGA -II [13]. Moslemi aplica MOPSO para sistemas de inventarios continuos [38].

Los modelos de inventario multi-objetivo tienen en cuenta la minimización de los costos y la maximización del nivel de servicio (minimizando la frecuencia y cantidad de productos no satisfechos con la demanda de los clientes), pero en un problema en el que se tienen varios productos comúnmente se debe decidir sobre las cantidades óptimas a ordenar de cada producto a un mismo proveedor teniendo en cuenta dos componentes [21]: Un Costo mayor de pedido independiente del número de productos en la orden y un Costo menor que depende

del número de diversos productos en la orden. Al problema anterior se le conoce como problema de reposición conjunta o "coordinada"(JRP, "joint replenishment problem"). Debido al costo de pedido mayor, si se realiza una reposición en grupo puede conducir a ahorros sustanciales en los costos, teniendo en cuenta un tiempo básico de ciclo y un número entero multiplicador de los periodos en los que se reabastece cada producto [30].

Para el problema de reposición conjunta Yousefi, et al. [63] plantean dos funciones a minimizar, los costos anuales esperados con la coordinación de los múltiples productos y el transporte inherente al pedido de productos. Para solucionar el problema multi-objetivo propuesto utilizan tres algoritmos el RAND, MOGA y la combinación de los dos anteriores, asumiendo una demanda anual determinista y uniforme en cada instante de tiempo.

Recientemente, Srivastav y Agrawal [54] realizan un estudio de la gestión de inventarios sobre productos de alta rotación aplicando Cuckoo Search multi-objetivo, en esta propuesta tienen en cuenta las funciones objetivo propuestas por Agrell [2], pero no tratan la coordinación del inventario . Mousavi et. al [39], presenta un modelo multi-producto, multi-objetivo para el control de inventarios, minimizando los costos(inflación) y el espacio utilizado de almacenaje, simultaneamente. Aplicaron tres algoritmos multi-objetivo para la solución el NSGA-II, NRGA, y MOPSO, en éste tampoco coordinan el inventario.

Dado el contexto anterior, la propuesta de este trabajo es plantear y programar un modelo que no sólo optimice el nivel de inventario (multi-producto) y nivel de servicio por producto, sino que tenga en cuenta una coordinación en el pedido y recibimiento de múltiples productos. Ya que el modelo permitiría disminuir costos anuales de mantenimiento, aumentar el nivel de servicio y disminuir los costos asociados a la coordinación del inventario.

1.2. Identificación del problema

Los sistemas de producción y distribución necesitan de una política de gestión en los inventarios debido a la necesidad inherente de velar por el cumplimiento del nivel de servicio y la minimización de costos asociados al pedido, almacenamiento, coordinación y transporte. La pregunta a responder en este trabajo es:

¿Cuál es la Política de Gestión de Inventarios que se debe implementar en un centro de fabricación y/o distribución, teniendo en cuenta la integración de todos los múltiples objetivos y restricciones en el sistema?

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar algoritmos bio-inspirados multi-objetivo para la optimización del control y coordinación de inventario multi-producto.

1.3.2. Objetivos específicos

- Construir un conjunto de funciones a optimizar, teniendo en cuenta los costos de almacenamiento, niveles de servicio, cantidad de pedidos desabastecidos, costos asociados al transporte y la coordinación del inventario.
- Comparar algoritmos bio-inspirados existentes que permita optimizar las funciones propuestas.
- Proponer criterios para el planteamiento de la mejor política de gestión de inventarios aplicando la técnica para la orden de preferencia por similitud con solución ideal (TOPSIS).

1.3.3. Organización del trabajo

De acuerdo a los objetivos, este trabajo se organiza de la siguiente forma:

En el **capítulo 2** se introduce a la toma de decisiones multicriterio, mostrando los conceptos de un problema de optimización y sobre la metodología seleccionada para la selección de alternativas, técnica para el ordenamiento preferencial por similitud con la solución ideal (TOPSIS).

En el **capítulo 3** se muestran los algoritmos bio-inspirados uni-objetivo y multi-objetivo aplicados en esté trabajo para la coordinación de múltiples productos. Para la parte Uni-Objetivo se desarrollan los algoritmos géneticos (AG), Enjambre de partículas (PSO), Cuckoo Search (CS) y Forrajeo de bacterias (BFOA). Mientras que para solución de problemas multi-objetivo se explicara el NSGA II ("nondominated sorting genetic algorithm II"), NS-PSO ("nondominated sorting particle Swarm Optimizer"), Cuckoo Search Muti-Objetivo (MOCS) y finalmente el BCMOA ("Bacterial chemotaxis multiobjetive optimization algorithm"). Finalmente se muestran las métricas de desempeño en los algoritmo uni-objetivo y multi-objetivo.

En el **capítulo 4** se explicara varios modelos de inventarios tales como el: EOQ, agregación de múltiples productos en una orden conjunta, modelo multi-objetivo con un solo

producto, modelos multi-objetivo para la agregación de múltiples productos. Finalmente se propone un **modelo multi-objetivo con múltiples productos**; buscando controlar y coordinar los pedidos para productos con demanda incierta, utilizando revisión continúa.

En el **capítulo 5** se dedica a presentar los resultados de aplica los algoritmos bio-inspirados propuestos a un conjunto de datos de medicamentos, el problema inicialmente se resuleve con algoritmos uni-objetivo para mostrar la contradicción entre objetivos y luego se aplican los multi-objetivo para encontrar una frontera de Pareto resultante. Finalmente se aplica TOPSIS y se comparan los algoritmos de acuerdo a las métricas de desempeño.

En el **capítulo 6** se dedica a presentar los resultados al aplicar los algoritmos bio-inspirados multi-objetivo a 5, 10, 20, 35 y 50 productos. Los parámetros de este capítulo son simulados de acuerdo a la distribución uniforme.

En el **capítulo 7** presenta las conclusiones y recomendaciones del trabajo.

2. Decisiones Multi-Objetivo: Conceptos básicos de Optimización y de selección de alternativas.

2.1. Introducción

En los problemas que involucra la toma de decisiones es factible encontrar varios objetivos que son contradictorios entre sí, pero sin importar la complejidad del problema y el nivel de contradicción entre objetivos el tomador de desiciones debera afrontar la selección y buscar la mejor alternativa. La toma de desiciones multicriterio (MCDM, Multi criterion decision making) consiste en dar un número finito de alternativas, donde cada alternativa tiene un desempeño en los criterios evaluados, y dadas estas alternativas el siguiente paso consiste en buscar las mejores alternativas para el tomador de la desición, clasificando las alternativas de acuerdo a su desempeño [55].

En este trabajo el entorno de la decisión multi-criterio, esta dado en dos etapas. En la primera etapa se generaran las fronteras no dominadas por medio de solución de problemas de optimización multi-objetivo (MOOP). y en la segunda etapa de acuerdo a unos pesos que reflejan la preferencia de cada objetivo, se aplica la técnica para el ordenamiento preferencial por similitud con la solución ideal (TOPSIS) con el fin de encontrar la mejor alternativa.

2.2. Conceptos básicos de Optimización

Dada la complejidad ligada a los problemas de las organizaciones, es necesario aplicar modelos matemáticos, asumiendo un sin número de supuestos y Paretociones que limitan la realidad propia del problema, pero nos permiten una proximidad óptima aceptable a los problemas a tratar [50].

Las técnicas de optimización han estado disponibles por más de un siglo. El primer acercamiento para encontrar esa "mejor"solución fue dado por Isaac Newton y Gottfried Leibniz en los años 1680s, quienes desarrollaron el cálculo diferencial en el cual por medio de funciones derivables se encuentran máximos y mínimos de funciones, lo cual abriría el camino a lo

que se conoce como optimización. En los años 1760s sería el matemático, físico y astrónomo JL Lagrange quien en una de sus obras en matemáticas habla acerca de los problemas de máximos y mínimos. En ella presentaba un nuevo método para resolver cierta clase de problemas de optimización. Lagrange aplicó el método de los multiplicadores para investigar el movimiento de una partícula en una superficie restringida.

En los 1840s aparecen A.L. Cauchy y C.F. Gauss quienes plantearon los métodos de optimización basados en gradientes. En los años 1900 H.L Gantt por medio de un gráfico programo eficientemente trabajos a hacer asignados en varias máquinas, conocido como el diagrama de Gantt. En 1915 Harris derivo una formulación matemática para la cantidad óptima de pedido para un producto a ordenar a un proveedor, conocido actualmente en el control y gestión de inventarios como EOQ ("Quantity order economic"). En 1917 Erlang propone la formulación de los sistemas de colas, analizando las longitudes y tiempos promedio de espera.

En 1939 William Karush en una tesis de maestría de la Universidad de California, plantea las condiciones mínimas necesarias de Paretociones con desigualdades. Posteriormente, en 1951,H. Kuhn y A. Tucker retomarían el estudio de W. Karush y plantearían las condiciones mínimas de desigualdades en modelos no lineales, llamadas como las condiciones KKT.

Durante la segunda guerra mundial, surge la necesidad de optimizar los recursos disponibles para el combate, naciendo la logística militar, en éste entorno aparecen conceptos como el problema de asignación de recursos, ruteo y el problema de la mochila. Estos serían inicialmente solucionados usando programación lineal y el método de simplex propuesta por G. Dantzig en 1947.

El auge de las primeras máquinas de computación permitió proponer algoritmos para la solución de problemas de optimización lineal en el año 1952. Muchas otras técnicas y heurísticas de optimización han nacido en la última mitad de siglo tales como: algoritmos genéticos, optimización por enjambre de partículas, forrajeo de bacterias, colonia de hormigas, entre otras. Las cuales son técnicas bio-inspiradas tratadas con mayor detalle en el capítulo "Algoritmos Bio-inspirados para la solución de problemas uni y multi objetivo".

2.3. Modelo Matemático

Una forma general de un modelo matemático puede ser representado como sigue [37, 50, 56]:

$$\Psi(\overrightarrow{\zeta_l}(\overrightarrow{x})) = [\zeta_1(\overrightarrow{x}), \zeta_2(\overrightarrow{x}), ..., \zeta_\lambda(\overrightarrow{x})]$$

Sujeto a:

$$g_i(x) \leq b_i \quad \forall i \in M_1 \subseteq M$$
$$h_i(x) = b_i \quad \forall i \in M \setminus M_1$$

$$x_j \leq 0 \quad \forall j \in N_1 \subseteq N$$
$$x_j \in \mathbb{R} \quad \forall j \in N \setminus N_1$$

Donde Ψ representa el sentido de optimización (maximización/minimización). ζ es la función objetivo $l = \{1, ..., \lambda\}$ con x variables. g_i representa las funciones de Paretoción del tipo desigualdad. Mientras que h_i las funciones de Paretoción de igualdad. Sea M el conjunto de Paretociones de igualdad y M_1 el conjunto de Paretociones de desigualdades, N=(1,..,j) el conjunto de variables x, y N_1 el conjunto de variables no negativas. Generalmente los b_i son constantes conocidas para modelos deterministas [50].

La optimización busca la mejor solución posible (punto /alternativa) dado un modelo mátematico. La cual se halla evaluando todas las posibles alternativas y seleccionando la de mejor desempeño en la función evaluada.

2.4. Clasificación de problemas de optimización

Los problemas de optimización pueden ser clasificados según: el número de funciones objetivo, la existencia de Paretociones, clasificación de las variables, clasificación de la función objetivo y la naturaleza de las variables tal como se muestra en la Figura **2-1**.

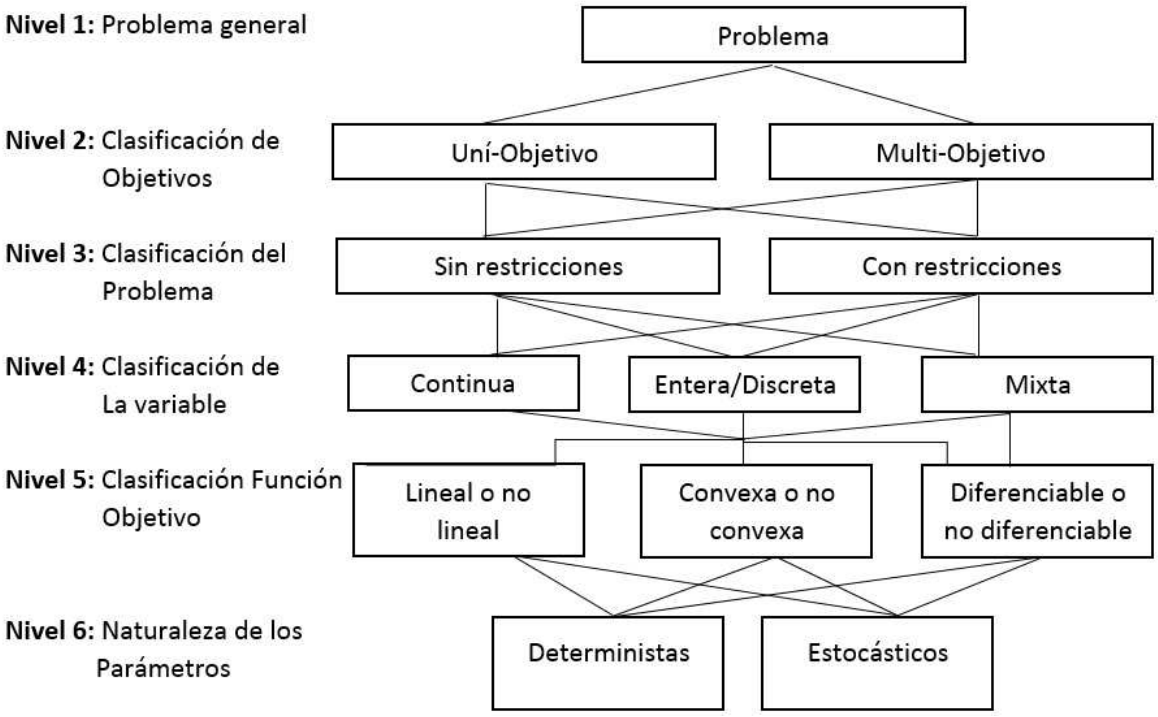

Figura 2-1.: Clasificación de problemas de optimización [50].

En el segundo nivel se tiene clasificación de acuerdo al número de funciones objetivos del problema y del tipo de objetivo (Minimizar, Maximizar). Normalmente en casos de problemas multi-objetivos las funciones objetivo se contradicen, característica que conduce a un conjunto de puntos óptimos conocidos como el **frente de Pareto** [6].

En el tercer nivel de la clasificación se tiene en cuenta si el problema es restringido/no restringido, es decir, el problema debe cumplir ciertas limitaciones dadas ya sea por los limites de las variables o de las funciones Paretotivas M. En el cuarto nivel se encuentra la naturaleza de las variables si son continuas $\mathbb{R}$ o discretas/enteras $\mathbb{Z}$, se pueden tener casos donde el problema tenga de los dos tipos por lo que será considerado como mixto.

Si la función objetivo es polinómica de primer grado será de tipo lineal, en caso contrario será del tipo no lineal (Funciones polinómicas de n grados, funciones trigonométricas, funciones de densidad de probabilidad, entre otras). La convexidad es considerada como una propiedad importante en la optimización clásica dado que muchos algoritmos y técnicas fueron desarrolladas basadas en la suposición que la función es convexa [50]. Si la matriz de las segundas derivadas parciales (Hessiana $\nabla^2\zeta_\lambda(x)$) de una función es definida positiva, se dice que la función es convexa. Por lo anterior es importante conocer si la función es diferenciable.

Finalmente si los párametros iniciales (entrada) producen invariablemente las mismas salidas el modelo es determinista.

2.5. Problemas de optimización Multi-Objetivo (MOOP)

La optimización multi-objetivo se define como un problema de encontrar un vector de variables x que satisfagan unas Paretociones M y optimicen $\Psi(\zeta_\lambda(x))$ [14]. Para encontrar los vectores con mejor desempeño existen varias técnicas de solución tales como: la programación lineal [11, 37], Escalarización y suma de pesos [16], programación por metas [15] y los algoritmos evolutivos [9]. En el **capítulo 3** se abordarán los algoritmos evolutivos en mayor profundidad.

2.5.1. Frontera óptima de Pareto

La optimalidad de Pareto es el concepto más importante en la optimización multi-objetivo, dado que al resolver un MOOP en general se llega a un conjunto de las soluciones no dominadas en el espacio de los objetivos y éste en contraparte en el espacio de decisión las cuales son llamadas las soluciones eficientes.

Definición: En un problema de minimización multi-objetivo dados los vectores de decisión $u = (u_1, u_2, ..., u_N)$ en el espacio de las decisiones. Se dice que u domina fuertemente a v

(denotado por $u \prec v$) si y sólo si $\forall i \in \{1, 2, ..., \lambda\}$,$\zeta_i(u) < \zeta_i(v)$. Se dice que u domina débilmente a un vector v ($u \preceq v$) si y sólo si $\forall i \in \{1, 2, ..., \lambda\}$, $\zeta_i(u) \leq \zeta_i(v)$ y $\exists i \in \{1, 2, ..., \lambda\}$, tal que $\zeta_i(u) \leq \zeta_i(v)$.

Una solución es dominante sobre otra si esta es superior en su desempeño en al menos uno de los criterios. Un conjunto de vectores decisión es el conjunto de los no dominados si no es miembro de los dominados por otro vector, tal como se muestra en el **Algoritmo 1** propuesto en [13] . La colección de todas las soluciones eficientes es llamado el conjunto eficiente $\Im_1$. La imagen del conjunto eficiente es referido a la frontera de Pareto (Figura **2-2**).

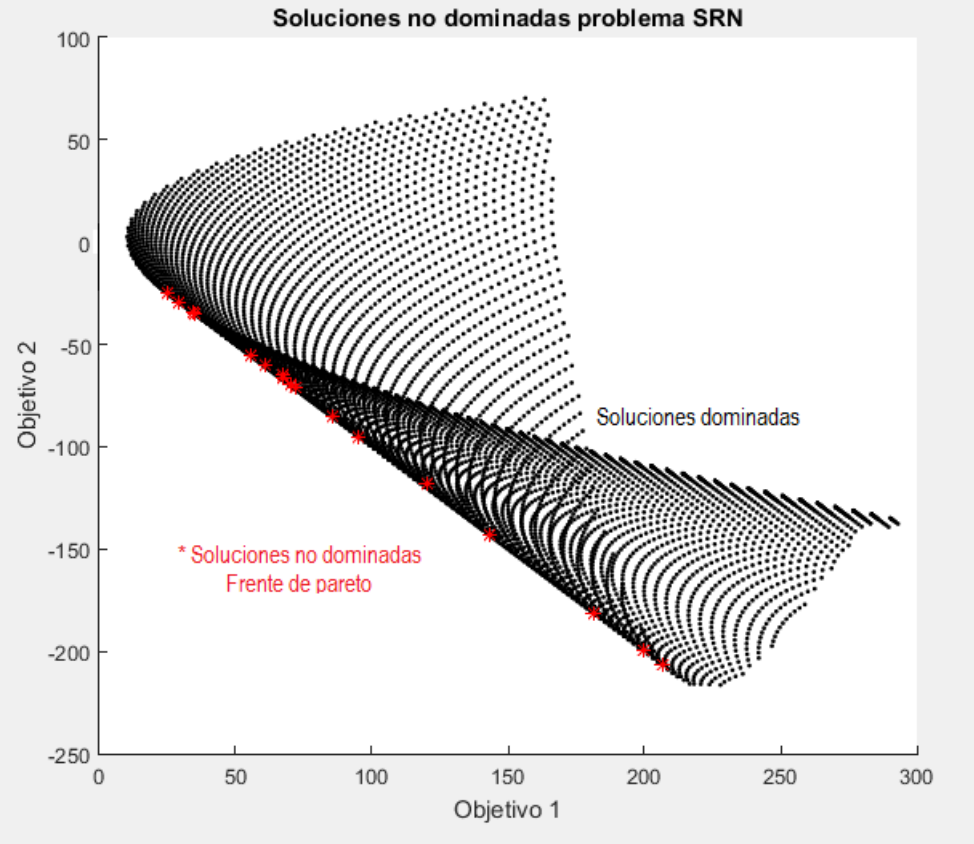

Figura 2-2.: Frontera de Pareto del problema SRN [13].

2.6. Técnica para el ordenamiento preferencial por similitud con la solución ideal (TOPSIS)

Esta técnica fue introducida por Hwang y Yoon [26]. La idea principal es elegir la alternativa más cercana a la solución ideal (solución óptima) y la más alejada de la solución ideal ne-

Algoritmo 1: Pseudo-Código Ordenamiento rápido de los no dominados [13]

```
for p ∈ ζ(X) do
    S_p = ∅;
    n_p = 0;
    for q ∈ ζ(X) do
        if p ≺ q then
            S_p = S_p ∪ {q};
        else
            if q ≺ p then
                n_p = n_p + 1;
    if n_p = 0 then
        ℑ_1 = ℑ_1 ∪ {p}
```

gativa (solución inferior). Luego la solución es aquella con la distancia euclidiana más corta desde la solución ideal y la distancia euclidiana más alejada de la solución ideal negativa [58].

Descripción procedimiento TOPSIS

Dado un conjunto de alternativas $A = \{A_k | k = 1, ..., n\}$, y un conjunto de criterios $\zeta = \{\zeta_j | j = 1, ..., \lambda\}$, donde $\xi = \{\xi_{k,j}, k = 1, ..., n; j = 1, ..., \lambda\}$ denota el conjunto de desempeños de la alternativa k en el criterio j y $w = \{w_j | j = 1, ..., \lambda\}$ es el conjunto de pesos [58]. Los pasos para ejecutar el algoritmo TOPSIS son:

Paso 1: Calcular la clasificación normalizada:

$$r_{k,j}(\xi) = \frac{\xi_{k,j}}{\sqrt{\sum_{k=1}^{n} \xi_{k,j}^2}} \tag{2-1}$$

Paso 2: Calcular la clasificación de pesos normalizados:

$$\upsilon_{k,j}(\xi) = w_j r_{k,j}(\xi) \tag{2-2}$$

Paso 3: Derivar el punto ideal positivo (PIS) y el punto ideal negativo (NIS):

$$PIS = A^+ = \{\upsilon_1^+(\xi), \upsilon_2^+(\xi), ..., \upsilon_j^+(\xi), \upsilon_\lambda^+(\xi)\} = \{(max_k \upsilon_{k,j}(\xi) | j \in J_1), (min_k \upsilon_{k,j}(\xi) | j \in J_2)\} \tag{2-3}$$

$$NIS = A^- = \{\upsilon_1^-(\xi), \upsilon_2^-(\xi), ..., \upsilon_j^-(\xi), \upsilon_\lambda^-(\xi)\} = \{(min_k \upsilon_{k,j}(\xi) | j \in J_1), (max_k \upsilon_{k,j}(\xi) | j \in J_2)\} \tag{2-4}$$

donde J_1 y J_2 son los beneficios y costos respectivamente.

Paso 4: Calcular la separación del PIS y NIS entre las alternativas. La separación puede ser medida usando la distancia euclideana:

$$D_k^* = \sqrt{\sum_{j=1}^{\lambda}[\upsilon_{k,j}(\xi) - \upsilon_j^+(\xi)]^2}, \forall k \tag{2-5}$$

$$D_k^- = \sqrt{\sum_{j=1}^{\lambda}[\upsilon_{k,j}(\xi) - \upsilon_j^-(\xi)]^2}, \forall k \tag{2-6}$$

Paso 5: Calcular la similaridad con PIS:

$$C_k^* = \frac{D_k^-}{D_k^* + D_k^-}, \forall k \tag{2-7}$$

Donde $C_k^* \in [0, 1] \forall k$

Paso 6: Ordenar de acuerdo a la similaridad con $PIS(C_k^*)$ y seleccionar el mejor.

3. Algoritmos Bio-inspirados para la solución de problemas uni y multi objetivo

3.1. Introducción

El hombre siempre ha tenido la necesidad de dar solución a una serie de problemas que aparecen como consecuencia o parte de su evolución. Estas soluciones dependen de su entorno, conocimientos y los recursos que se encuentren disponibles, siempre tratando de encontrar la "mejor"solución posible para sus problemas. Gracias a esta búsqueda el hombre ha desarrollado nuevos campos del conocimiento como los algoritmos inteligentes, los cuales apoyan la toma de decisiones.

La inteligencia artificial busca diseñar e implementar en sistemas computacionales la realización de actividades y tareas consideradas como inteligentes [49] cuanto trata de imitar las habilidades humanas o de otros seres para solucionar problemas en el dominio de conocimiento establecido en el momento, entendiendo que este dominio es dinámico, incierto e inespecífico [44]. En la inteligencia artificial, existe una rama de algoritmos para la optimización los cuales están basados sobre la teoría de la evolución, conocidos como algoritmos evolutivos [4]. Estos algoritmos se han aplicado en muchos dominios del conocimiento en los últimos años, teniendo una especial variedad de aplicaciones desde la ingeniería y la ciencias de la computación a la industria, ecología, sociología y medicina [6].

Los algoritmos evolutivos, al igual que la inteligencia artificial, tienen como característica principal el imitar el comportamiento de la naturaleza, de la interacción social o de eventos físicos [9]. Tratando de mimetizar este comportamiento en las poblaciones ya sea en su manera de reproducirse y/o conservarse, o en el comportamiento social por enjambre de las especies. Uno de los algoritmos evolutivos basados en la reproducción/conservación de las especies es el algoritmo genético [20, 25], en el que cada individuo de la población tiene un desempeño, el cual le permite, reproducirse o no, realizando un cruce de genes con otro individuo. Además de tener una probabilidad de mutación en sus genes.

Otro de los algoritmos basados en la conservación de las especies es el de búsqueda de Cuco (Cuckoo search) [60, 61], en el cual una especie de ave utiliza una estrategia parasitaria para su conservación. El ave imita los huevos de otras especies y los intercambia en los nidos, aprovechándose del cuidado del ave hospedera. El cuco tiene una probabilidad de ser descubierto, por lo cual sus huevos serán desechados por el hospedero. El algoritmo consiste en maximizar la probabilidad del que el huevo

no sea abandonado, cada pajaro realiza los movimientos por medio de vuelos de Levy en el espacio de búsqueda.

En cuanto a los algoritmos evolutivos inspirados en el comportamiento social por enjambré, han mostrado efectividad para solucionar problemas multi-objetivo. Uno de los algoritmos más destacados es optimización por enjambré de partículas (PSO) propuesto por Kennedy y Eberhart [28]. Una partícula (por ejemplo las abejas) representa una posible solución en el espacio de búsqueda. La mejor posición anterior (dada por la función de desempeño) de la partícula i-esima se registra y la mejor partícula entre todas las partículas en la población es tenida en cuenta como el punto de referencia global. Las partículas buscan la mejor solución personal y global encontrada hasta el momento a una velocidad del cambio de posición para cada partícula. Permitiendo que estas partículas converjan a la mejor posición dada por la función de desempeño.

Siguiendo por la misma línea de inspiración de algoritmos basados en enjambres, Passino propone el algoritmo de forrajeo de bacterias (BFOA, "Bacterial Foraging Optimization Algorithm")[43], en el que un grupo de bacterias E. coli buscan nutrientes. Cada una de las bacterias se comunica con las demás por medio del envió de señales. El proceso es que cada bacteria realiza pequeños movimientos mientras que busca nutrientes, es llamado quimiotaxis, lo cual es mimetizado por el algoritmo realizando este proceso en el espacio de búsqueda[12].

Los algoritmos mencionados anteriormente, fueron planteados inicialmente para resolver problemas de optimización uni-objetivo, pero posteriormente fueron adaptados para resolver problemas multi-objetivo. El algoritmo "Elitist Non-dominated Sorting Genetic Algorithm"NSGA-II [13], el NSPSO [34], Cuckoo Search Multiobjetive [62] y BCMOA [23] son explicados para mostrar las diferentes estrategias utilizadas en la conformación de frentes de Pareto.

En lo que sigue de este capítulo se resumirá y se mostrara el pseudo-código los algoritmos tanto uni-objetivo y multi-objetivo. Finalmente se muestran las medidas de desempeño que serán utilizadas en este trabajo para realizar comparaciones.

3.2. Algoritmos bio-inspirados Uni-Objetivo

3.2.1. Algoritmos genéticos (AG)

Los algoritmos genéticos fueron propuestos por Holland (1975)[20, 25], en éstos se tienen una población de cromosomas (con información de cada variable, gen) que evolucionan en un número dado de interacciones (generaciones), en las cuales se buscan que una población de hijos "mejore"de acuerdo a una función de desempeño, para este fin de se utilizan tres operadores básicos: selección, cruce y mutación. La selección tiene como objetivo premiar a los mejores cromosomas para su supervivencia, aunque puede dar oportunidad de que pasen cromosomas no tan buenos. El cruce permite crear nuevos hijos de acuerdo al emparejamiento de padres, esto permite explorar el espacio de búsqueda y finalmente la mutación, la cual permite con pequeños cambios en un cromosoma

explotar el espacio de búsqueda.

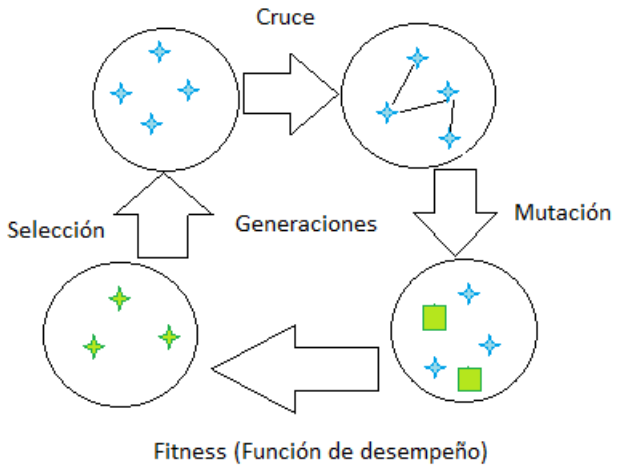

Figura 3-1.: Esquema de los algoritmos genéticos.

Inicialmente se debe generar una población de cromosomas aleatoriamente, permitiendo diversificación en el espacio de búsqueda. Luego de generada la población inicial se evalua cada individuo (cromosoma) en la función de desempeño ("fitness"). En cada generación el número de cromosomas permanece constante.

Definiciones y operadores en algoritmos genéticos

Cromosomas: Son el vector de las variables de decisión (cada variable es un gen del cromosoma), donde cada vector representa una posible solución del problema. Estas son representadas por medio de una cadena de bits ya sean binarias, reales o enteras. En algunos casos se realizan cambios de números reales y enteros a binarios y viceversa.

Población: Es el número de cromosomas en el espacio de búsqueda.

Generación: Es el número de iteraciones en los que se genera una nueva población.

Función de desempeño: Mide qué tan bueno o malo en cada uno de los cromosomas por medio de una función objetivo.

Selección: En este operador se seleccionan los individuos (cromosomas) con los mejores desempeños, dandole las ventaja de ser escogidos como padres de la próxima generación de modo que sus genes "superiores"sean transmitidos.

Existen varias formas para seleccionar los padres, siendo las más comunes:

- **Torneo:** n individuos son escogidos al azar. El mejor de esos n es seleccionado.

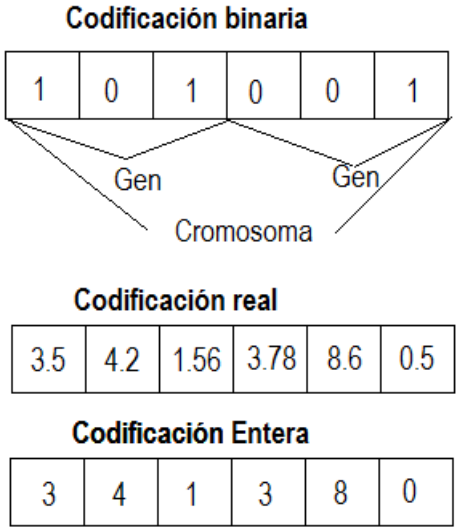

Figura 3-2.: Tipo de colificaciones en algoritmos géneticos.

- **Rueda de ruleta:** A cada cromosoma de la población se le asigna una porción de la rueda de ruleta, proporcional a su valor de adecuación.
- **Elitismo:** Los mejores cromosoma de la población son escogidos.

Cruce: Es el mecanismo en el se comparten los genes entre los padres seleccionados, procreando la misma cantidad de hijos.

- **Punto simple:** Un solo punto de cruce es seleccionado aleatoriamente, todos los bits después de este punto, son cambiados.
- **2 o n puntos:** 2 o n puntos de cruce son seleccionados aleatoriamente. El segmento del cromosoma entre los 2 o n puntos es cambiado.
- **Uniforme:** Seleccionando cada gen aleatoriamente de cada padre con igual probabilidad. Cada gen es independiente de los otros y del punto de cruce.

Mutación: Aleatoriamente se selecciona un gen de un cromosoma y se modifica su valor. Para cada gen se genera un número aleatorio entre $[0, 1]$ si este es menor a la probabilidad de mutación el gen modifica su valor.

Algoritmo 2: Pseudo-Código AG

```
1 Generar una población inicial;
2 Calcular la función de desempeño de cada cromosoma;
3 for g ← 1 to Generaciones do
4     for r ← 1 to Reproducciones = Población/2 do
5         Seleccionar los cromosomas a hacer padres;
6         Cruzar los cromosomas seleccionados como padres para crear los hijos;
7         Mutar los genes de los cromosomas hijos;
8         Calcular la función de desempeño de cada cromosoma;
9 Visualizar resultados;
```

3.2.2. Optimización por enjambre de partículas (PSO)

El PSO ("Particle Swarm Optimization") propuesto por Kennedy y Eberhart [28] es un método heurístico que evoca los movimientos de grandes bandadas de peces y grupos de insectos. Consiste en estudiar los movimientos de cada individuo del grupo en busca de un objetivo pero cuenta con una forma de comunicación con el resto del grupo. Cada uno de los integrantes del grupo se mueve impulsado por tres factores. Primero, su propia experiencia de los mejores lugares en los que ha estado hasta el momento. Segundo, los mejores lugares que han sido descubiertos por otro miembro del grupo. Tercero, una componente de inercia del movimiento anterior que amortigüe las velocidades de las partículas. Cada una de las partícula representa una posible solución a un problema (espacio de búsqueda).

$$x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$$

donde x_i representa la particula i-ésima $i \in I$ y $n \in N$ es el número de variables del problema. La mejor posición anterior (Memoria o nostalgía) de la partícula i-ésima se registra y se representa como $P_i = \{P_{i1}, P_{i2}, ..., P_{in}\}$. La mejor partícula entre todas las partículas (mejor posición histórica) en la población es representada por g. La velocidad del cambio de posición (velocidad) para cada partícula es representada como $V_i = \{V_{i1}, V_{i2}, ..., V_{in}\}$. Las partículas son iteradas k-veces, cada una de las partículas de la población recorre el espacio de soluciones con una velocidad V_i hacia nuevas posiciones X_i, de acuerdo con su propia experiencia P_i, y con la experiencia aportada por el mejor de sus congéneres, G.

$$V_{in}(k+1) = \omega V_{in}(k) + c_1 r_1 (P_{in}(k) - x_{in}(k)) + c_2 r_2 (g(k) - x_{in}(k)) \qquad \forall i, n \tag{3-1}$$

$$x_{in}(k+1) = x_{in}(k) + \chi V_{in}(k+1) \qquad \forall i, n \tag{3-2}$$

Donde ω, es el coeficiente de inercia, si este es grande facilita la exploración global, mientras que un peso menor de inercia tiende a facilitar la explotación local en el espacio de búsqueda. C_1 es la aceleración cognitiva y C_2 la aceleración social. R_1 y R_2 son números aleatorios uniformemente distribuidos entre [0,1]. Cada particula es actualizada en cada dimensión del problema, donde χ es el factor constricción.

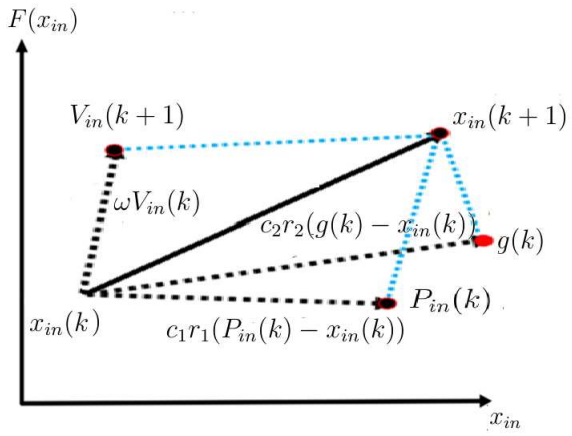

Figura 3-3.: Mecanismo de búsqeda del PSO.

Las ventajas primordiales del PSO son, es un algoritmo simple de implementar, y puede alcanzar soluciones en espacios de búsquedas complejos, en poco tiempo dado una alta velocidad de convergencia (donde las partículas buscan la mejor posición global y propia). Además tiene pocos parámetros para ajustar. PSO se ha aplicado con éxito: optimización, la formación de redes neuronales artificiales, control de sistema difuso, entre otras áreas. Algunas desventajas del PSO son: si la mejor solución está estancada en algún óptimo local, todas las partículas tenderán rápidamente a concentrarse en ese punto [1]. Otra importante desventaja era su poco control para tener un balance entre la exploración y explotación, para contrarrestar estos problemas, se utiliza el coeficiente de inercia W [51]. El PSO está enfocado en la optimización de funciones continuas no lineales. Sabiendo que las variables del modelo propuesto son continuas. A continuación se muestra el pseudo-código utilizado en este trabajo:

Algoritmo 3: Pseudo-Código PSO

Inicialización de partículas;
$x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$;
$V_i = \{V_{i1}, V_{i2}, ..., V_{in}\}$;
$P_i = X_i$;
$G = X_i$;
for $k \leftarrow 1$ **to** K **do**
    for $i \leftarrow 1$ **to** I **do**
        for $n \leftarrow 1$ **to** N **do**
            Actualizar posición y velocidad;
            $V_{in}(k+1) = \omega V_{in}(k) + c_1r_1(P_{in}(k) - x_{in}(k)) + c_2r_2(g(k) - x_{in}(k))$;
            $x_{in}(k+1) = x_{in}(k) + \chi V_{in}(k+1)$;
    Calcular el valor de la función objetivo $F(x_{in})$;
    Encontrar la mejor posición personal $P_{in}(k)$;
    Encontrar la mejor posición global $g(k)$;

3.2.3. Cuckoo Search (CS)

El Cuckoo Search (CS) es un algoritmo de optimización desarrollado por Xin-She Yang y Suash Deb en 2009 [60, 61]. Fue inspirado por el parasitismo de algunas especies de cuco poniendo sus huevos en los nidos de otras aves de acogida (de otras especies). Si un ave descubre los huevos que no son suyos, abandona su nido y construye un nuevo nido en otro lugar. Algunas especies de cuco tales como la Tapera han evolucionado de tal manera que los cucos femeninos son a menudo muy especializados en el mimetismo en colores y el patrón de los huevos.

La Búsqueda cuco (CS) utiliza las siguientes representaciones: cada huevo en un nido representa una solución (Figura **3-4**). El objetivo es utilizar las nuevas y potencialmente mejores soluciones (cucos) para reemplazar una no tan buena solución en los nidos. En la forma más simple, cada nido tiene un huevo.

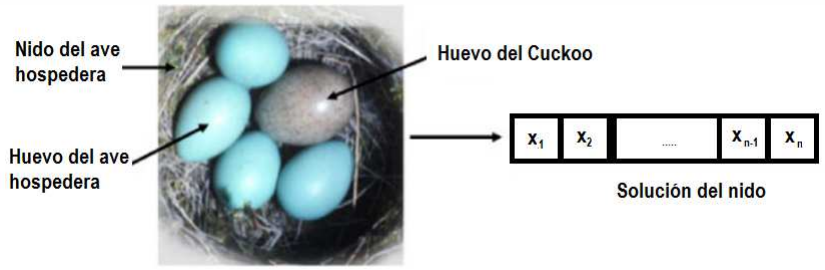

Figura 3-4.: Representación de soluciones en cuckoo search.

En la naturaleza, los animales buscan la comida de una manera aleatoria o cuasi aleatoria. En general, la trayectoria de alimentación de un animal es efectivamente un paseo aleatorio porque el siguiente movimiento se basa en la ubicación actual y la probabilidad de transición a la siguiente ubicación. En qué dirección se elige depende implícitamente en una probabilidad que puede modelarse matemáticamente [61]. Muchos animales e insectos tienen un vuelo parecido a la distribucción de Lévy. La cual es una caminata aleatoria, en general es una cadena de markov (Estado/Ubicación), la cual solo depende de la ubicación actual [61].

$$x_{t+\tau} = x_t + \Phi(t)$$

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\lambda)$$

$$Levy = t^{-\lambda},\ (1 < \lambda \leq 3)$$

Reglas Cuckoo Search

- Cada Cuco pone un único huevo cada vez, y lo deja en un nido elegido al azar.
- Los mejores nidos con mayor calidad del huevo pasarán a la siguiente generación.

- El número de nidos disponibles en cada generación está predefinido, y los huevos serán descubiertos y descartados con una probabilidad $p_a \in [0, 1]$. Por facilidad una fracción p_a del total de los nidos será sustituida por nidos nuevos cada generación.
- El pseudo-código del algoritmo Cuckoo Search es presentado en el Algoritmo 4. Donde x_i es el i-esimo $i \in I$ nido y $n \in N$ el número de variables del problema.

Algoritmo 4: Pseudo-Código Cuckoo Search

```
Generar población de n nidos hospederos x_i (i = 1, ..., I);
Función objetivo f(x), x = (x_1, x_2, ..., x_n)^T;
for k ← 1 to K do
    Seleccionar un cuckoo (digase, i) al azar por medio de los vuelos de Lévy;
    Evaluar la calidad/desempeño F_i;
    Calcular el valor de la función objetivo F(i);
    Seleccionar un nido i entre I (digase , j) aleatoriamente ;
    if F_i > F_j then
        Reemplazar j como la nueva solución;
    Abandonar una fracción P_a de los peores nidos;
    Construir nuevos nidos en las nuevas ubicaciones por medio de vuelos de Lévy;
    Mantener las mejores soluciones (nidos con el mejor desempeño) ;
    Clasificar las soluciones y buscar la mejor solución actual ;
Visualizar resultados;
```

3.2.4. Algoritmo de optimización por Forrajeo De Bacterias (BFOA)

El algoritmo de optimización por forrajeo de bacterias fue propuesto en el año 2002 por Passino [43], en el cual se mimetiza el proceso completo de forrajeo de las bacterias E. Coli: Chemotaxis (movimientos de giro y nado), reproducción y eliminación-dispersión.

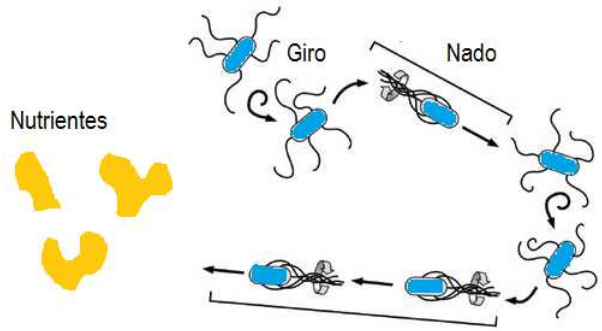

Figura 3-5.: Movimientos de las bacterias E. COli.

Las bacterias realizan dos movimientos básicos el giro(Voltereta) y el nado para la búsqueda de nutrientes. Las bacterias (E. Coli) se mueven hacia los puntos de concentración de nutrientes (dentro

de esta generación ocurre una eliminación y dispersión), algunas bacterias que se acercan a puntos de sustancias nocivas son eliminadas. Después ocurre la reproducción y posteriormente casi todas las bacterias se colocan en puntos de concentración de nutrientes. Cuando las bacterias han encontrado los puntos de concentración de nutrientes, aunque no todos estos puntos son óptimos, por lo cual las bacterias se agrupan, utilizando la comunicación mediante segregación de atrayentes y realizan la búsqueda juntas. Finalmente, las bacterias encuentran el punto más alto de concentración de nutrientes, donde de nuevo son dispersadas para encontrar otras soluciones. El modelo cuenta con los siguientes párametros:

- **p:** Número de variables.
- **S:** Número de bacterias.
- **Nc:** Número de veces que la población de bacterias podrá nadar o girar.
- **Ns:** Número de veces en que la bacteria podrá nadar.
- **Nre:** Número de reproducciones.
- **Ned:** Número de eliminaciones/dispersiones.
- **Ped:** probabilidad de eliminación/dispersión.

Algoritmo 5: Pseudo-Código BFOA

for $l \leftarrow 1$ **to** Ned **do**
  for $k \leftarrow 1$ **to** Nre **do**
    for $j \leftarrow 1$ **to** Nc **do**
      for $i \leftarrow 1$ **to** Nc **do**
        Computar $J(i,j,k,l)$. Dado $J(i,j,k,l) = J(i,j,k,l) + J_{cc}(\theta^i(j,k,l), P(j,k,l))$ dado la atracción entre bacterias J_{cc};
        Guardar $J_{last} = J(i,j,k,l)$ para encontrar un mejor costo a través de la corrida;
        Realizar un giro (voltereta) generando un vector aleatorio $\Delta(i) \in \Re^p$ con cada elemento $\Delta_m(i), m = 1,2,...,p$ un número aleatorio en $[-1,1]$;
        Nadar,$\theta^i(j+1,k,l) = \theta^i(j,k,l) + \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$;
        Calcular $J(i,j+1,k,l) = J(i,j+1,k,l) + J_{cc}()\theta^i(j+1,k,l), P(j+1,k,l))$;
        while $m < Ns$ **do**
          if $J(i,j+1,k,l) < J_{last}$ **then**
            $J_{last} = J(i,j+1,k,l)$;
            $\theta^i(j+1,k,l) = \theta^i(j+1,k,l) + \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$;
          else
            $m = Ns$;
    $J^i_{health} = \sum_{j=1}^{Nc+1} J(i,j,k,l)$;
  Con probabilidad Ped, eliminar-dispersar cada bacteria;
Visualizar resultados;

3.3. Algoritmos bioinspirados Multi-Objetivo

3.3.1. NSGA II (Elitist Non-dominated Sorting Genetic Algorithm)

Este algoritmo fue propuesto por Deb y algunos de sus estudiantes [13]. En este algoritmo mejoran las falencias mostradas por NSGA [53], en el cuál la complejidad computacional era mayor para ordenar los frentes no dominantes $O(MN^3)$, no aplicaba elitismo para la selección de los individuos y necesitaba definir el párametro σ_{share} el cuál tiene una gran influencia en los resultados del algoritmo.

El algoritmo de NSGA II propone una técnica de ordamiento de los frentes de Pareto de una manera más rápida disminuyendo la complejidad computacional a $O(MN^2)$. Parte de este procedimiento se mostro en el **Capítulo 2** "*Código Ordenamiento rápido de los no dominados*"en este capítulo se completará para los demás frentes. También el algoritmo implemento la utilización de una técnica de crowding que no requiere especificar parámetros para la preservación de diversidad en la población, eliminando la dependencia de σ_{share} utilizado por el NSGA original. A continuación se detallan los pasos utilizados por el NSGA-II :
Paso 1: Crear una población inicial aleatoria P_0 de N individuos. Evaluar las funciones objetivo.
Paso 2: Ordenar la población, basados en los criterios de no dominancia.

Inicializar con la primera solución q, la cuál es almacenda en S_p el cuál es el conjunto de soluciones no dominadas. A partir de la segunda solución, cada solución p es comparado por dominancia con cada una de las soluciones que pertenecen a S_p. Si una solución p domina a cualquier solución q de S_p esa solución es removida de S_p; pero si una solución q de S_p domina a p esta es ignorada. Si una solución p no es dominada por ninguna solución q, esa solución es almacenada en S_p. Asimismo una población S_p aumenta solo por la unión de soluciones no dominadas [22].

El mismo precedimiento se repite para las soluciones restantes y se conforma un segundo conjunto de soluciones no dominadas $\Im_2$. El procedimiento continua hasta cuando todas las soluciones hallan sido clasificadas.

Paso 3: Atribuir a cada solución un valor de adaptación igual a su nivel de dominancia. Para garantizar el mismo potencial reproductivo de todas ellas.

Paso 4: Aplicar los procedimientos de selección por torneo binario, cruce y mutación para crear una nueva población Q_0. Unir ambas poblaciones en un conjunto R_0 de tamaño $2N$. Los cuatro pasos anteriores constituyen un procedimiento inicial, para las siguientes generaciones. El algoritmo trabaja con R_t y Q_t

Paso 5: Formar una población conjunta R_t conformada por $P_t \cup Q_t$. Ordenar la población conjunta de acuerdo a los criterios de no dominancia obtener el conjunto de soluciones no dominadas.

Paso 6: Conformar una nueva población P_{t+1} tomadas de los dos conjuntos de soluciones no

Algoritmo 6: Pseudo-Código Ordenamiento rápido de los no dominados en NSGA-II [13]

```
for p ∈ ζ(X) do
    S_p = ∅;
    n_p = 0;
    for q ∈ ζ(X) do
        if p ≺ q then
            S_p = S_p ∪ {q};
        else
            if q ≺ p then
                n_p = n_p + 1;
    if n_p = 0 then
        ℑ_1 = ℑ_1 ∪ {p};
    i = 1;
    while ℑ_i ≠ ∅ do
        Q = ∅;
        foreach p ∈ ℑ_i do
            foreach q ∈ S_p do
                n_q = n_q − 1;
                if n_q = 0 then
                    q_rank = i + 1;
                    Q = Q ∪ q
        i = i + 1;
        ℑ_i = Q
```

dominadas, comenzando con llenar a P_{t+1} con el conjunto $\Im_1$, luego $\Im_2$ y asi sucesivamente. Si el tamaño de $\Im_1$ es menor que N, todas las soluciones que pertenecen a $\Im_1$ pasan a P_{t+1}; la cantidad de individuos que falten para completar N individuos en P_{t+1} es seleccionada de los niveles restantes de dominancia de acuerdo a dos criterios de ordenamiento: El valor de adaptación y la clasificación por distancia de hacinamiento.

Paso 7: Aplicar selección por torneo binario, cruzar y mutar para los individuos de P_{t+1} y generar una nueva generación de Q_{t+1}.

Paso 8: Si aún no se ha iterado el número máximo de iteraciones ir al paso 5.

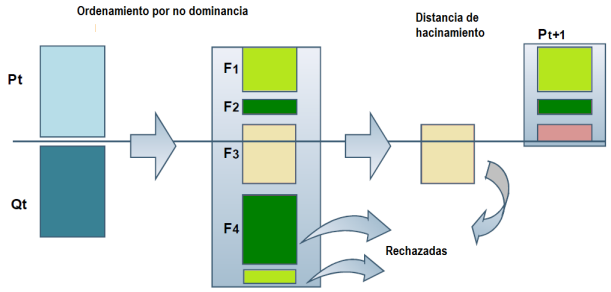

Figura 3-6.: Procedimiento de NSGA II.

3.3.2. NSPSO (Non-dominated Sorting Particle Swarm Optimizer)

El NSPSO fue un algoritmo proopuesto por Li en el año 2003 [34], donde extendió los principios básicos del PSO para hacer uso del mejor histórico de cada particula, $P_{in}(k)$. Por otra parte aplicó los conceptos de Ordenamiento rápido de los no dominados y la distancia de hacinamiento propuesto en el NSGA-II.

Algoritmo 7: Pseudo-Código NSPSO [34]

```
Inicializar la población y almacenar la población en PSOList;
Evaluar cada particula en la población ;
for k ∈ iteraciones do
    Identificar las particulas No Dominadas y almacenarlas en la lista nonDomPSOList;
    Calcular para cada particula el conteo de nicho y la distancia de hacinamiento ;
    Recurrir al nonDomPSOList según a el conteo de nicho y la distancia de hacinamiento ;
    for i ← 1 to Cantidad de particulas do
        Seleccionar aleatoriamente la mejor global g(k) para la i-ésima particula desde
          nonDomPSOList;
        Calcular la nueva velocidad
          V_in(k+1) = ωV_in(k) + c1r1(P_in(k) − x_in(k)) + c2r2(g(k) − x_in(k))∀i, n;
        Calcular la nueva posición x_in(k+1) = x_in(k) + χV_in(k+1)∀i, n;
        Añadir a la i-ésima particula P_i y la nueva X_i es una población temporal, almacenar en
          nextPopList. Observe que; P_i y X_i ahora coexisten. Note que nextPopList ahora es de
          tamaño de 2N.;
    Identificar las particulas que son no dominadas desde nextPopList, y almacenar esto en
      nonDomPSOList. Las particulas que son dominadas son almacendas en nextPopListRest.;
    El PSOList debe estar vacío para la siguiente iteración;
    Seleccionar aleatoriamente los miembros de nonDomPSOList y añadir esto a PSOList;
    while PSOListsize < Cantidad de particulas do
        Identifique las particulas no dominadas desde nonDomListRest y almacene en
          nextNonDomList.;
Calcular métricas de desempeño;
```

Para mayor información del algoritmo consultar [34].

3.3.3. MOCS (Multi-Objetive Cuckoo Search)

El algoritmo de búsqueda de cuckoo multi-objetivo fue propuesto por Yang y Deb en el año 2013 [62]. En este algoritmo modifican las reglas propuestas por ellos para CS.

Reglas de MOCS

- Cada cuco pone K huevos a la vez, y los vuelca en un nido elegido al azar. El huevo k corresponde a la solución de la función objetivo de orden k.
- Los mejores nidos con mayor calidad del huevo pasarán a la siguiente generación.
- Cada nido será abandonado con un probabilidad Pa y un nuevo nido con huevos K se construirá.

Algoritmo 8: Pseudo-Código MOCS

```
Generar población de n nidos hospederos x_i(i = 1, ..., I) con K huevos;
Función objetivo f_1(x), ..., f_k(x), ∀x = (x_1, x_2, ..., x_n)^T;
for t ← 1 to Generaciones do
    Seleccionar un cuckoo (digase, i) al azar por medio de los vuelos de Lévy;
    Evaluar si éste está en la frontera óptima ℑ_1 ;
    Seleccionar uno de los n nidos aleatoriamente (denótese con j);
    Evaluar las K soluciones del nido j;
    if las nuevas soluciones del nido j dominan a las del nido i then
        Reemplazar el nido i por el conjunto de soluciones nuevas del nido j;
    Abandonar una fracción P_a de los peores nidos;
    Construir nuevos nidos en las nuevas ubicaciones por medio de vuelos de Lévy;
    Mantener las mejores soluciones (nidos con el mejor desempeño) ;
    Clasificar las soluciones y buscar las soluciones de frente de Pareto actuales ;
Visualizar resultados;
```

3.3.4. BCMOA (Bacterial Chemotaxis Multiobjetive Optimization Algorithm)

El algoritmo de optimización multi-objetivo de forrajeo de bacterias fue propuesto por Guzmán en el año 2009 [23, 22]. Este algoritmo usa el procedimiento de Ordenamiento rápido de los no dominados, comunicación entre los miembros de la colonia y una estrategía simple para el cambio de posiciones de las bacterias, lo cual permite explorar el espacio de búsqueda. El algoritmo puede ser resumido en los siguientes pasos [23]:

Paso 1: Definir los parámetros S (número de bacterias), p número de variables, CHS_{max} (Máximo

número de pasos chemotaxicos), i (Indice de pasos chemotaxicos) y k (índice para parámetros). Inicializar la población de bacterias; $j = 1$. La primera posición de cada bacteria $\theta^i(j) \in \Re^p$ es inicializada en un número aleatorio real dentro del rango especificado de las variables de desición.
a. Almacene todas las bacterias $b^i(j)$ y su posición actual $\theta^i(j)$ en la lista Bac.
b. Actualizar los parámetros $LT_k(j)$, $ST_k(j)$ y $SW_k(j)$, acorde a:

$$Fac(j) = (CHS_{max} - j)/CHS_{max} \tag{3-3}$$

$$LT_k(j) = \frac{1}{S}(max(\Im_1(j))_k - min(\Im_1(j))_k)Fac(j) \tag{3-4}$$

$$ST_k(j) = 0{,}1LT_k(j) \tag{3-5}$$

$$SW_k(j) = (\theta_k(j)_{fuertes} - \theta_k(j)_{debiles})Fac(j) \tag{3-6}$$

Paso 2: Para cada bacteria $b^i(j)$ en la lista Bac, evaluar $J(\theta^i(j))$ que es el costo (Valor de la función objetivo) en la actual posició y almacenar el valor en la lista ObjFunVal.
Paso 3: Clasifique la ubicación de todas las bacterias usando el procedimiento de Ordenamiento rápido de los no dominados en $J(\theta(j))$. Almacene cada bacterias que halla sido clasificada en $\Im_1(j)$, y almacene todas las bacterias dominadas en la lista Bacdom.
Paso 4: Para cada bacteria en $\Im_1(j)$, genere un vector aleatorio $\Delta^i(j) \in \Re^p$ en el que cada elemento $\Delta^i_k, k = 1, 2, ..., p$ es un núnero aleatorio entre $[-1, 1]$.
Paso 5: Para cada bacteria en $\Im_1(j)$, compare las actuales valores de las funciones objetivo $J(\theta^i(j))$ con el previo $J(\theta^i(j))_{prev}$ almacenado en $ObjFunVal_{prev}$ por el concepto de no dominancia. Para $j = 1, ObjFunVal_{prev} = ObjFunVal$. La nueva ubicación para la bacteria es dado por:
- Si $\theta^i(j)_{prev}$ domina a $\theta^i(j)$, $\theta^i_k(j+1) = \theta^i_k(j)_{prev} + ST_k(j)\Delta^i_k(j)$
- Si $\theta^i(j)$ domina a $\theta^i(j)_{prev}$, $\theta^i_k(j+1) = \theta^i_k(j) + ST_k(j)\Delta^i_k(j)$
- Si ambos $\theta^i(j)$ y $\theta^i(j)_{prev}$ son no dominados, $\theta^i_k(j+1) = \theta^i_k(j) + LT_k(j)\Delta^i_k(j)$
Paso 6: Para cada bacteria débil $\theta^i(j)$ en Bacdom, seleccionar aleatoriamente una bacteria fuerte $\theta^i(j)_{fuertes}$ desde $\Im_1(j)$,
$\theta^i_k(j+1) = \theta_k(j)_{fuertes} + \theta_k(j)_{fuertes} r1_k + SW_k(j)$, donde $r1_k$ es un número aleatorio entre $[-0{,}01, 0{,}01]$.
Paso 7: Aplicar la estrategía de pared absorbente [47].
Paso 8: $ObjFunVal_{prev} = ObjFunVal$.
Paso 9: Si $j \leq CHS_{max}, j = j + 1$ y vuelva a paso 1 item a, en otro caso vaya a 10.
Paso 10: Detenga el algoritmo.

En [22] se detalla el procedimiento.

3.4. Medidas de desempeño

Para poder comparar que un algoritmo es más eficiente que otro, se utilizarán las siguientes métricas para el caso de algoritmos Uni-objetivo y Multi-objetivo.

3.4.1. Métricas algoritmos Uni-Objetivo

Tiempo Computacional: El tiempo CPU en segundos utilizado en ejecutar el algoritmo.
Mejor desempeño: El mejor desempeño F_j encontrado en el algoritmo j en todas las replicas $R \in Rep$.

$$F_j = min\{F_j(R)\} \tag{3-7}$$

Promedio del desempeño: El promedio del desempeño en el algoritmo j de todas las replicas:

$$\overline{F_j} = \frac{\sum_{R=1}^{|Rep|} F_j(R)}{|Rep|} \tag{3-8}$$

Desviación del desempeño: Calcula la desviación estándar del mejor desempeño en cada replica del algoritmo j.

$$\sigma_j = \sqrt{\frac{1}{|Rep|-1}\sum_{R=1}^{|Rep|}(F_j(R) - \overline{F_j})^2} \tag{3-9}$$

3.4.2. Métricas algoritmos Multi-Objetivo

Tiempo Computacional: El tiempo CPU en segundos utilizado en ejecutar el algoritmo.
Indicador de dominancia de Pareto [19]: Dado S_j las soluciones obtenidas por el algoritmo j y $B = \{b_i | \forall b_i, \nexists a_i \in (\bigcup_j S_j) \prec b_i\}$ donde $a_i \prec b_i$ indica que a_i domina a b_i. Está métrica mide la proporción de las soluciones no dominadas contribuidas por S_j de todas las soluciones no dominadas.

$$NR(S_j) = \frac{|S_j \cap B|}{|B|}, \forall j \tag{3-10}$$

Diversidad [13]: Defina:

$$\Delta(S_j) = \sum_{i=1}^{|S_j|-1} \frac{(d_i - \overline{d})}{|S_j| - 1}, \forall j \tag{3-11}$$

donde d_i es la distancia euclideana entre soluciones consecutivas, y $\overline{d}$ es el promedio de d_i. Cuando la distancia de todas las soluciones son iguales, entonces $d = d_i$, implicando que $\Delta(S_j) = 0$ y tiene una perfecta distribución.

4. Modelos de optimización para la gestión y control de inventarios multi-producto

Para poder satisfacer las demandas a tiempo, se espera que la compañía tenga un inventario a la mano. Los principales objetivos para la buena administración de los inventarios es la minimización de los costos asociados con mantener el inventario y maximizar la satisfacción del cliente. Por lo que los modelos de inventario deben responder las siguientes preguntas:

- ¿Cuándo se debe realizar el pedido de un producto?
- ¿Qué tan grande debe ser cada pedido?

Este capítulo explicara el **Modelo de lote económico de pedido EOQ** (Economic orden Quantity) el cual fue propuesto por Harris [24] en el que se asume un solo producto y una función de costo a ser minimizada (uni-objetivo). Del modelo EOQ han surgido varias mejoras como la **agregación de múltiples productos en una orden conjunta** [7] en el que los minoristas o proveedores en un solo pedido permiten una reducción en el tamaño del lote de productos individuales, ya que los costos fijos de ordenar y de transporte se reparten entre varios productos. El modelo de agregación de múltiples productos en una orden conjunta inicialmente minimiza el costo de la gestión de los inventarios pero no tiene en cuenta el costo de transporte. En Youssefi se proponen un **modelo Multiobjetivo para la agregación de múltiples productos** en una orden conjunta para la minimización de los costos de gestión del inventario (costos de mantener y costos de pedir conjuntamente) y a la vez tiene en cuenta un costo de transporte independiente del pedido [63]. Hasta ahora en los modelos se supone que la demanda es conocida por lo que la política está determinada por las condiciones de los parámetros iniciales.

Para poder aproximarse un poco más a la realidad se han propuesto modelos con demanda incierta por lo que ya se tendrían políticas que dependen del comportamiento estocástico de los parámetros iniciales. Agrell [2] propone un modelo con comportamiento de demanda estocástica para un producto en el cual se minimizan tres funciones objetivo: El costo total, la frecuencia esperada de faltantes y la cantidad de faltantes; este **modelo multi-objetivo con un solo producto** no tiene en cuenta la coordinación de pedido para varios productos.

Finalmente se propone un **modelo multi-objetivo con múltiples productos**; buscando controlar y coordinar los pedidos para productos con demanda incierta, utilizando revisión continúa.

4.1. Costos relacionados con los modelos de inventario

Costo de pedido y organización: Es un costo fijo que no depende del tamaño del pedido. Este costo de pedido u organización incluirá el costo de trabajo administrativo y facturación asociada con un pedido si este proviene de una fuente externa. Si el producto se realiza internamente en la empresa el costo de mano de obra para la preparación y detención de una máquina se incluirá en el costo de pedido y organización.

Costo de compra unitario: Este es el costo variable asociado con la compra de una sola unidad. Por lo general este costo incluye el costo de mano de obra variable, el costo fijo variable y el costo de materia prima asociada con la compra o producción de una sola unidad. Si los bienes se piden a una fuente externa, el costo de compra unitario debe incluir el costo de envió.

Costo de retención o posesión: Es el costo de mantener una unidad de inventario durante un periodo. Este costo incluye costo de almacenaje, costo de aseguramiento, impuestos al inventario y el costo debido a la posibilidad de descomposición o robo. Uno de los principales componentes del costo de inventarios es el costo de oportunidad en el que se incurre al invertir capital en inventario. Normalmente las empresas asumen entre un 20 a 40 porciento del costo de compra unitario, sin olvidar el efecto de las economías de escala.

Costo de escasez o agotamiento de existencias: Este costo se causa cuando un cliente pide un producto y en ese momento no se satisface; son los costos asociados a la perdida de oportunidad por la no satisfacción de la demanda. Dentro de este se puede tener en cuenta la perdida de ventas potenciales de futuros clientes, utilidades dejadas de percibir, pagar salarios extras o comprar los productos a la competencia. Hay que tener en cuenta que algunos clientes permiten posponer el pedido por lo que tendríamos pedidos pendientes, en caso contrario ventas perdidas.

4.2. Suposiciones del modelo de cantidad económica de pedido EOQ

Pedido repetitivo: La decisión de pedido es repetitiva dado de que se pide en una forma regular.

Demanda constante: Se supone que la demanda ocurre a una tasa constante en un intervalo de tiempo determinado, y se asume que ésta demanda es conocida.

Plazo de entrega constante: El plazo de entrega de cada pedido es una constante conocida, L. Por plazo de entrega se entiende el tiempo transcurrido entre el instante cuando se hace un pedido y el instante de arribo.

Pedidos continuos: Un pedido se puede hacer en cualquier instante. Los modelos que permiten esto se llaman modelos de revisión continua. Si la cantidad de inventario disponible se revisa de manera periódica y los pedidos se pueden hacer sólo cada cierto tiempo, se tiene un modelo de

revisión periódica.

4.3. Modelo básico de lote económico de pedido

De acuerdo al modelo clásico de lote económico de pedido (EOQ), se requiere que se cumplan las siguientes suposiciones:

- La demanda es determinística y ocurre a una tasa constante.
- Si se hace un pedido de cualquier tamaño (por ejemplo, Q unidades), se incurre en un costo K de pedido y organización.
- El tiempo de reposición de cada pedido es cero ($L = 0$).
- No se permite escasez.
- El costo por unidad-año de inventario de reserva es h.

Se define D como el número de unidades demandas por año. Dadas las anteriores suposiciones, el modelo EOQ determina una política de pedidos que reduce la suma anual de costos de pedido, compra y retención [24]:

$$\Psi(\Theta(Q)) = \frac{D}{Q}K + cD + \frac{Q}{2}h \tag{4-1}$$

Donde $\frac{D}{Q}K$ es el costo de pedido, cD es el costo de compra teniendo en cuenta el costo de compra por unidad c por la demanda D, y $\frac{Q}{2}$ es el inventario promedio por el costo de almacenar h. Se debe encontar un Q que minimice Θ, y este es llamado Q^*. Con este Q^* se encuentra la cantidad de pedidos, $n = \frac{D}{Q^*}$, y la frecuencía esperada, $t = \frac{Q^*}{D}$. Este modelo lo podemos clasificar como:

- **Determinista** dado que la demanda es constante y conocida.
- **Uni-producto** el modelo sólo tiene en cuenta un producto.
- **Uni-Objetivo** el costo de pedido, compra y retención es la única función objetivo.
- **Sin coordinación** ya que solo tiene en cuenta un producto.

4.4. Tamaño del lote con múltiples productos

El control de varios artículos del inventario tiene problemas que no se presentan en los modelos de artículos individuales de inventario.
Dado que los costos de colocación, transporte y recepción de pedidos crecen con la diversidad de los productos por lo que puede ser más económico recibir un camión que contiene un solo producto que recibir un camión con muchos productos diferentes, ya que el esfuerzo de actualización y reabastecimiento del inventario es mucho menor para un solo producto. Una fracción del costo fijo de un

pedido puede estar relacionada con el transporte (esto depende sólo de la carga y es independiente de la variedad del producto en el pedido). Una fracción del costo fijo se relaciona con la carga y la recepción (este costo aumenta con la variedad en el camión). Ahora, estudiaremos qué tamaño óptimo del lote puede establecerse en esa situación. El objetivo es llegar a los tamaños de lote y política de ordenar que minimicen el costo total [7].

Notación:
D_i: Demanda anual del producto i
S: Costo de ordenar, costo en el que se incurre cada vez que se hace un pedido, independiente de la variedad de productos incluidos en el pedido.
s_i: Costo adicional de ordenar en que se incurre si el producto i se incluye en el pedido.

Para este tipo de problemas se puede considerar tres estrategías:
1. Cada producto es solicitado de manera independiente, es decir no existe una **coordinación** en la recepción de pedidos
2. Los productos se solicitan en forma conjunta (**coordinación**), todos los productos son ordenados-entregados en cada lote de pedido.
3. Los productos se ordenan de modo conjunto, pero no todos los pedidos contienen todos los productos; esto es, cada lote abarca un subconjunto de los productos dentro del portafolio de la compañia.

La primera estrategía no utiliza la agregación, lo que da como resultado altos costos. El segundo, agrega todos los productos en cada pedido. Su debilidad radica en que los productos de baja demanda se agregan con los de alta demanda en cada pedido. Esta agregación completa genera altos costos si el costo específico de ordenar productos de demanda baja es muy elevado. En tal situación, es preferible ordenar productos de baja demanda con menos frecuencia que los de demanda alta. Esta práctica provoca una reducción del costo específico de ordenar un producto de baja demanda. Como repercusión, es probable que la tercera estrategía da lugar a un costo menor. Sin embargo, es más complejo de coordinar [7].

El problema de pedir varios productos de manera conjunta es conocido en la literatura como JRP ("joint replenishment problem") [31, 29, 63]. Las suposiciones básicas de este modelo son similares a las de EOQ, demanda deterministica y uniforme, los faltantes no son permitidos, no hay descuentos por cantidad y el costo de almacenamiento es lineal, a continuación definiremos la notación usada:

Notación:
T: Tiempo entre reposiciones sucesivas (años)
S: Costo de ordenar en que se incurre cada vez que se coloca un pedido, independiente de la variedad de productos incluidos en el pedido. (Costo de ordenar en que se incurre cada vez que se hace un pedido ($/pedido)
ϱ: Costo total anual (Costos de almacenamiento y costos de pedido) para todos los productos ($/año)

i: 1, 2, . . . , n, indice de producto, n total de productos
D_i: Demanda anual para el producto i (unidades/año)
h_i: Costo anual de almacenar el producto i ($/unidades/año)
s_i: Costo adicional de ordenar en que se incurre si el producto i se incluye en el pedido. ($/pedido)
Q_i: Cantidad a ordenar del producto i
T_i: Intervalo de tiempo entre pedidos sucesivos del producto i.

Para definir los costos, se tiene que tener en cuenta las estrategías mencionadas anteriormente. Si se tiene en cuenta la primera estrategía se incurriria en un mayor costo, por lo que no es tenida en cuenta para el manejo de múltiples productos. La segunda estrategía en la literatura se conoce como DGS ("Direct grouping strategy") y la tercera estrategía como IGS(Ïndirect grouping strategy"). La literatura sugiere que IGS supera a DGS debido a tener un mayor costo de ordenar en que se incurre cada vez que se coloca un pedido, y que muchos productos pueden ser repuestos de manera conjunta utilizando IGS [17]. Al utilizar IGS el tiempo de ciclo para cada producto es múltiplo entero k_i del ciclo de reposición tiempo T. Por lo tanto, el tiempo de ciclo para el producto i es:

$$T_i = k_i T \tag{4-2}$$

y la cantidad a ordenar para el producto i es:

$$Q_i = T_i D_i = k_i T D_i \tag{4-3}$$

Costo total de inventario:

$$C_H = \sum_{i=1}^{n} \frac{Q_i h_i}{2} = \frac{T}{2} \sum_{i=1}^{n} k_i D_i h_i \tag{4-4}$$

El costo total anual de pedido es:

$$C_o = \frac{S}{T} + \sum_{i=1}^{n} \frac{s_i}{k_i T} = \frac{S + \sum_{i=1}^{n} \frac{s_i}{k_i}}{T} \tag{4-5}$$

por lo que el costo total a minimizar es:

$$\Psi(\varrho(T, k_i)) = \frac{T}{2} \sum_{i=1}^{n} k_i D_i h_i + \frac{S + \sum_{i=1}^{n} \frac{s_i}{k_i}}{T} \tag{4-6}$$

En este modelo se tiene en cuenta **múltiples productos**, con una demanda **deterministica** y una reposición conjunta o **coordinada**, pero sólo tiene una función objetivo la cúal no tiene en cuenta el costo de transporte. En [3] proporcionan una prueba de que el JRP es un problema NP-hard, por lo tanto , es poco probable que exista un algoritmo de tiempo polinómico para resolver el problema. Hasta ahora los modelos mostrados sólo tenian en cuenta una función objetivo, en la siguiente sección se muestra la extensión a modelos multi-objetivo.

4.5. Modelos de inventario Multi-Objetivo

En esta sección analizaremos los modelos de inventario de revisión continua que tienen en cuenta más de una función objetivo.

4.5.1. Modelo Multi-objetivo con múltiples productos, con demanda constante.

El modelo propuesto por [63] tiene en cuenta múltiples productos y a la vez la coordinación del inventario, en este se utiliza una estrategía IGS pero añade una nueva función objetivo para minimizar el costo de transporte.

Supuestos

- Los parámetros son determinísticos y conocidos.
- No se permite escasez.
- No se permite descuentos en cantidad.
- El costo de almacenamiento es lineal y es calculado basado en el promedio de inventario almacenado.
- La estrategía para el agrupamiento es IGS ("indirect grouping strategy")
- El costo de transporte es basado en el máximo inventario almacenado y no hace parte del costo del inventario.

Funciones Objetivo:

$$\Psi(\varrho(T,k_i)) = \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{1}{T}(S + \sum_{i=1}^{n}\frac{s_i}{k_i}) \tag{4-7}$$

$$\Psi(\Delta(T,k_i)) = T\sum_{i=1}^{n} tr_i k_i D_i \tag{4-8}$$

Sujeto a:

$$T_i = k_i T \tag{4-9}$$

$$Q_i = D_i T_i\,,\ \forall i = 1,2,....,n \tag{4-10}$$

$$T \geq 0, k_i \geq 0 \tag{4-11}$$

En el modelo se minimiza el costo total de pedido y costo de inventario y tiene en cuenta la minimización del costo de transporte del inventario. Para poder aproximarse más a la realidad se han propuesto modelos con demanda incierta por lo que ya se tendrían políticas que dependen del comportamiento estocástico de los parámetros iniciales.

Este modelo tiene en cuenta **múltiples productos**, con una demanda **deterministica** y una reposición **coordinada**, teniendo la función de costo del modelo JRP adicionando una función objetivo para el costo de transporte, por lo cual es un modelo **multi-objetivo**.

4.5.2. Modelo multi-objetivo mono-producto y demanda incierta

En este modelo de optimización para un producto se trata de minimizar el costo total, frecuencia esperada de ocasiones de desabastecimiento anuales y el número esperado de productos faltantes anualmente [2].

Parámetros:

- **D:** Valor esperado de la demanda anual
- **A:** Costo de ordenar
- **c:** Costo unitario del articulo
- **h:** Tasa de mantener inventario
- $f_{DL}(x)$: Variable aleatoria que representa la demanda durante el tiempo de entrega, con media μ_L y desviación σ_L

Variables de desición:

- Q: Cantidad de pedido óptima
- **SS:** factor de seguridad, Permite tener una protección contra la incertidumbre de la demanda

Funciones Objetivo:

$$\Psi(\phi(Q, SS)) = \frac{AD}{Q} + hc(\frac{Q}{2} + k\sigma_L) \tag{4-12}$$

$$\Psi(\varphi(Q, SS)) = \frac{D}{Q} \int_r^{\infty} f_{DL}(x)dx \tag{4-13}$$

$$\Psi(\chi(Q, SS)) = \frac{D}{Q} \int_r^{\infty} (x - r)\, f_{DL}(x)dx \tag{4-14}$$

Sujeto a:

$$0 \leq Q \leq D$$
$$0 \leq SS \leq \frac{D}{\sigma_L}$$

Este modelo tienen en cuenta:

- Una demanda **estocástica**
- Es **multi-objetivo**
- **Sin coordinación** debido a que este modelo esta propuesto para **un producto**.

4.6. Modelo Propuesto: Multiobjetivo con múltiples productos, con demanda estocástica. (Coordinación de pedidos)

En esta sección se propone un modelo multi-objetivo, teniendo en cuenta los costos de almacenamiento, niveles de servicio(cantidad y frecuencia de pedidos de desabastecidos), costos asociados al transporte y la coordinación del inventario.

En el modelo **Modelo multiobjetivo con un solo producto y demanda incierta**, hay una función del costo total donde se involucra un factor de seguridad (SS), este factor de seguridad se utiliza para tener en cuenta una demanda aleatoria durante el lapso de reabastecimiento. En Youssefi [63] no se tenía en cuenta este factor de seguridad, debido a que asumen una demanda constante y no hay pedidos pendientes (desabastecimiento).

Para nuestro modelo se propone que la demanda de cada producto se comporte estocásticamente con una distribución de probabilidad normal $f_{Di}(x)$ de media $\mu_{Di} = E(D_i)$ y desviación $\sigma_{D_i} = \sqrt{var(D_i)}$, de igual manera se tiene en cuenta la demanda durante el lapso de reabastecimiento para cada producto, la cual es una distribución de probabilidad normal $f_{DLi}(x)$ con media μ_{DLi} y desviación σ_{DLi}. El tiempo de entrega de cada producto L_i se puede comportar de dos maneras [59]:
Constante:

$$\mu_{DLi} = L_i * E(D_i) \ \forall i \tag{4-15}$$

$$\sigma_{DLi} = \sigma_{Di}\sqrt{L_i} \ \forall i \tag{4-16}$$

Variable: Se asume un L_i variable con media $\mu_{L_i} = E(L_i)$ y desviación σ_{L_i}.

$$\mu_{DLi} = E(L_i) * E(D_i) \ \forall i \tag{4-17}$$

$$\sigma_{DLi} = \sqrt{E(L_i) * \sigma_{Di}^2 + E(D_i)^2 * \sigma_{L_i}^2} \ \forall i \tag{4-18}$$

Una actividad esencial para cualquier organización, es determinar el tamaño del pedido y su frecuencia, esto para establecer mecanismos de gestión y control. Uno de los mecanismos más usados son los modelos de revisión continua (r,Q) en el que una orden de tamaño Q es pedida cuando la posición del inventario está por debajo del punto de reorden r [52].

La determinación de (r,Q) depende del tiempo de entrega y las fluctuaciones de la demanda, donde el objetivo es maximizar el nivel de servicio y minimizar los costos totales de inventario [56]. Agrell [2] estableció tres objetivos sobre costo y desabastecimiento para planificar dos variables de control, el tamaño de pedido Q y el factor de seguridad SS que es un término del punto de pedido r ($r_i = SS_i\sigma_{DLi} + \mu_{DLi}$)[5]. Dado $p_u > SS_i$: Probabilidad de una variable aleatoria normal (0,1) tome un valor de SS_i o más grande.

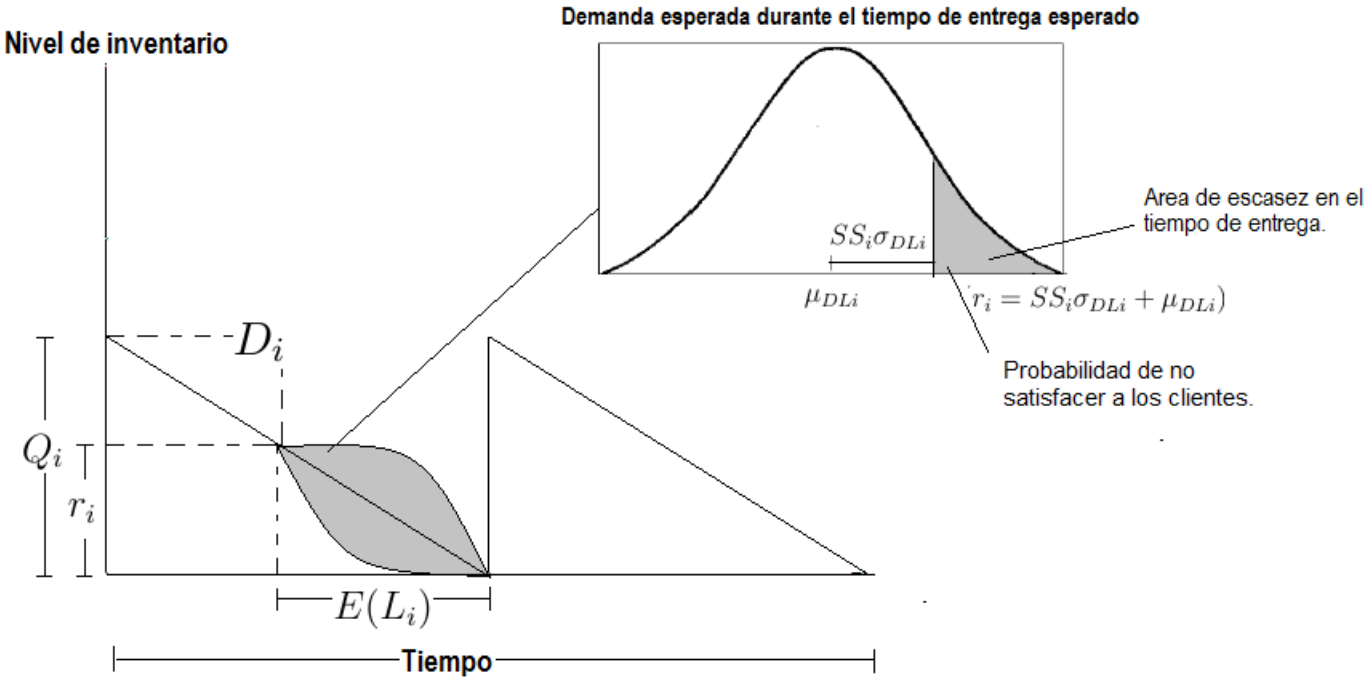

Figura 4-1.: Comportamiento en el tiempo de los niveles de inventario de productos coordinados.

En la Figura **4-1** se ve la interpretación de $p_u > SS_i$.

Se puede expresar que $p_u > SS_i$ es a menudo expresado como $1 - \Phi(SS_i)$, donde $\Phi(\cdot)$ es la función de distribución acumulada de la función normal estándar.

Dado que las variables del modelo JRP son T y k_i; y las de los modelos de inventarios estocáticos son Q_i y SS_i, se buscara proponer dos problemas equivalentes:

Modelo I. En el cual se dejaran la ecuaciones en términos de las variables Q_i, SS_i y k_i.

Modelo II Las ecuacuiones se plantean en términos de T, SS_i y k_i. Sabiendo que $T_i = k_i T$ y $Q_i = E(D_i)T_i$

Notación:

Conjunto:
i : Producto Observado
Parámetros:
h_i : Costo de almacenaje anual para el producto i.
$E(D_i)$: Valor esperado de la demanda anual para el producto i.
σ_{Di} : Desviación de la demanda esperada anual para el producto i.
S : Costo mayor de ordenar.
S_i : Costo de pedido menor para el producto i.
$E(L_i)$: Valor esperado del tiempo de reposición para el producto i.
σ_{L_i} : Desviación del tiempo de reposición esperada para el producto i.
CB_i : Costo de escasez o agotamiento de existencias para el producto i.
tr_i : Costo de transportar el producto observado i.
μ_{DLi} : Demanda durante el lapso de reabastecimiento para cada producto i.
σ_{DLi} : Desviación de la demanda durante el lapso de reabastecimiento para cada producto i.

Modelo I

Variables de decisión:
Q_i : Cantidad a ordenar del producto i.
r_i : Punto de reorden del producto i.
SS_i : Factor de seguridad del producto i.
k_i : Múltiplo entero del ciclo de reposiciones sucesivas de tiempo en el producto i.

Funciones Objetivo:
Función de costo de inventario y pedido coordinado

$$(f_1)\ \Psi(\delta(Q_i, k_i, SS_i)) = \left(\sum_{i=1}^{n}\left(\frac{Q_i}{2} + SS_i\sigma_{DLi}\right)h_i\right) + \left(\sum_{i=1}^{n}\frac{E(D_i)}{Q_i}(Sk_i + s_i)\right) \tag{4-19}$$

Función de frecuencia esperada de ocasiones de desabastecimiento anuales.

$$(f_2)\ \Psi(\vartheta(Q_i, SS_i) = \sum_{i=1}^{n}\frac{E(D_i)}{Q_i}(1 - \Phi(SS_i)) \tag{4-20}$$

Función de la cantidad esperada de productos faltantes anualmente.

$$(f_3)\ \Psi(\varphi(Q_i, SS_i) = \sum_{i=1}^{n}\left[\frac{E(D_i)\sigma_{DL_i}}{Q_i}(\phi(SS_i) - SS_i(1 - \Phi(SS_i)))\right] \tag{4-21}$$

Función de costo de transporte del inventario

$$(f_4)\ \Psi(\nu(Q_i, k_i)) = \sum_{i=1}^{n} tr_i Q_i \tag{4-22}$$

Restricciones

$$0 \leq Q_i \leq D_i \tag{4-23}$$

$$0 \leq SS_i \leq \frac{D_i}{\sigma_{DLi}} \ \forall i \tag{4-24}$$

$$T_i = \frac{Q_i}{D_i} \ \forall i \tag{4-25}$$

$$T = \frac{T_i}{k_i} \forall i \rightarrow \frac{T_1}{k_1} = \frac{T_2}{k_2} = ... = \frac{T_n}{k_n} \tag{4-26}$$

$$T \geq 0, \ SS_i \geq 0, \ k_i \in \mathbb{Z}^+ \tag{4-27}$$

$$r_i = SS_i \sigma_{DLi} + \mu_{DLi} \tag{4-28}$$

Modelo II

Variables de decisión:
T : Tiempo entre reposiciones sucesivas (años).
r_i : Punto de reorden del producto i.
SS_i : Factor de seguridad del producto i.
k_i : Múltiplo entero del ciclo de reposiciones sucesivas de tiempo en el producto i.

Funciones Objetivo:
Función de costo de inventario y pedido coordinado

$$(f_1) \ \Psi(\delta(T, k_i, SS_i)) = \left(\sum_{i=1}^{n} \left(\frac{T k_i E(D_i)}{2} + SS_i \sigma_{DLi} \right) h_i \right) + \frac{1}{T} \left(S + \sum_{i=1}^{n} \frac{s_i}{k_i} \right) \tag{4-29}$$

Función de frecuencia esperada de ocasiones de desabastecimiento anuales.

$$(f_2) \ \Psi(\vartheta(T, k_i, SS_i)) = \frac{1}{T} \sum_{i=1}^{n} \frac{1}{k_i} (1 - \Phi(SS_i)) \tag{4-30}$$

Función de la cantidad esperada de productos faltantes anualmente.

$$(f_3) \ \Psi(\varphi(T, k_i, SS_i)) = \frac{1}{T} \sum_{i=1}^{n} \left[\frac{\sigma_{DL_i}}{k_i} (\phi(SS_i) - SS_i(1 - \Phi(SS_i))) \right] \tag{4-31}$$

Función de costo de transporte del inventario

$$(f_4) \ \Psi(\nu(T, k_i)) = T \sum_{i=1}^{n} E(D_i) t r_i k_i \tag{4-32}$$

Restricciones

$$T_{min} \leq T \leq T_{max} \tag{4-33}$$

$$T_{min} = min_{i \in n} \left(\frac{2 s_i}{h_i E(D_i)} \right)^{1/2} \tag{4-34}$$

$$T_{max} = \left[\frac{2\,(S + \sum_{i=1}^{n} s_i)}{\sum_{i=1}^{n} E(D_i)h_i}\right]^{1/2} \tag{4-35}$$

$$k_{imin} \leq k_i \leq k_{imax}\ \forall i \tag{4-36}$$

$$k_{imin} = 1/T_{max}\ \forall i \tag{4-37}$$

$$k_{imax} = 1/T_{min}\ \forall i \tag{4-38}$$

$$0 \leq SS_i \leq 3{,}9\ \forall i \tag{4-39}$$

$$Q_i = T_i * E(D_i)\ \ \forall i \tag{4-40}$$

$$T_i = T * k_i\ \ \forall i \tag{4-41}$$

$$T \geq 0,\ \ SS_i \geq 0,\ \ k_i \in \mathbb{Z}^+ \tag{4-42}$$

$$r_i = SS_i\sigma_{DLi} + \mu_{DLi} \tag{4-43}$$

El número de variables del problema dependerá de la cantidad de productos a coordinar (n), En el modelo I tendriamos $3 * n$, mientras que en el modelo II serán $(2 * n) + 1$. Por lo que el modelo II tendra menos variables a tratar reduciendo la dimensionalidad del problema. En el modelo II no se tiene que cumplir con la restricción: $T = \frac{T_i}{k_i}\,\forall i \rightarrow \frac{T_1}{k_1} = \frac{T_2}{k_2} = ... = \frac{T_n}{k_n}$ la cual es una restricción muy compleja de cumplir, a cambio se agregaron otras restricciones que permiten cumplir la restricción pero son de más baja complejidad. Por todo lo anterior, el modelo II será el utilizado para resolver el problema **Multiobjetivo con múltiples productos, con demanda estocástica, coordinando la solicitud de productos**. En el Anexo A se detalla como se contruyó cada una de las funciones objetivo.

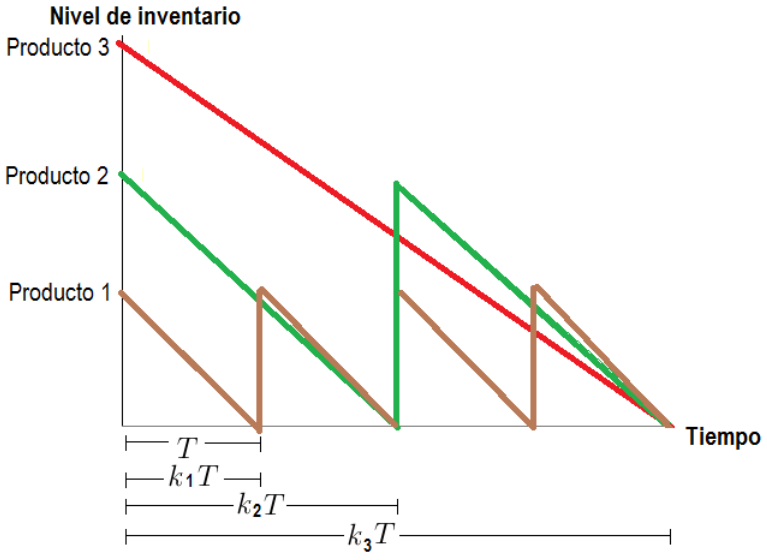

Figura 4-2.: Comportamiento en el tiempo de los niveles de inventario de productos coordinados con una estrategía indirecta.

5. Aplicación de algoritmos bioinspirados para el control, coordinación y gestión de inventario de medicamentos

En este capítulo se realizara un análisis del modelo propuesto para cada una de las cuatro funciones objetivo de manera separada, con el fin de entender el comportamiendo individual de cada una y poder explicar los posibles conflictos que pueden existir al aplicarlas de manera conjunta en un problema multi-objetivo.

Se aplicaran los algoritmos genéticos (AG), optimización por enjambre de partículas (PSO), cuckoo search (CS) y el algoritmo de optimización por Forrajeo De Bacterias (BFOA). Para cada uno de los algoritmos se realizaron 100 réplicas, de 100 iteraciones cada una. Los parámetros propios de cada algoritmo se muestran en la Tabla **5-1**. Los algoritmos se compararan de acuerdo a las métricas de tiempo computacional, mejor desempeño del algoritmo en las réplicas, promedio del desempeño por réplicas y desviación del desempeño por réplicas.

Luego el problema se resolvera mediante algoritmos bio-inspirados multi-objetivos tales como NSGA II, NSPSO, MOCS y BCMOA para obtener la frontera óptima de Pareto de acuerdo a los parámetros seleccionados de los algoritmos (Tabla **5-7**). Las soluciones encontradas se comparan mediante métricas como el indicador de dominancia de Pareto, diversidad y tiempo computacional; obteniendo una frontera de Pareto con aportes de todos los algoritmos. Finalmente se aplicó TOPSIS seleccionando el punto más cercano a la solución ideal. Los algoritmos fueron ejecutados en un ordenador con Procesador FX 6300, 8 GB DDR3 1600 Corsair y una Tarjeta video R7260x2G DDRS.

Se obtuvo un conjunto de datos sobre el consumo de medicamentos en un centro médico de Chile [18]. A continuación se detalla como se calcularon los parámetros requeridos por el modelo II propuesto.
h_i : Para cada producto i se conoce su volumen v_i (cm^3), peso w_i (gr) y valor comercial c_i ($), dado lo anterior se calculo el indice por volumen IV_i y peso IW_i para estimar el indicador logistico IL_i [33].

$$IV_i = \frac{v_i}{\sum_{i=1}^{n} v_i} \tag{5-1}$$

$$IW_i = \frac{w_i}{\sum_{i=1}^{n} w_i} \tag{5-2}$$

$$IL_i = \alpha * IV_i + \beta * IW_i \tag{5-3}$$

$$\alpha = \beta = 0{,}5 \tag{5-4}$$

Donde α y β serán los números con los que se pondera la importancia de cada uno de los dos factores indicados, peso y volumen respectivamente. Los cuales deben cumplir que $\alpha + \beta = 1$. Luego de calcular IL_i se multiplica por su respectivo valor comercial, dando como resultado h_i.

$E(D_i)$: Se conoce demanda mensual para cada producto i.

σ_{Di} : Se conoce demanda mensual para cada producto i, por lo que al suponer normalidad será igual a $\sigma_{Dmensual_i} * \sqrt{12}$.

S : Se realizó un estimativo del costo promedio de operación del vehiculo, costo de operarios involucrados y los costos directos del viaje (utilización del vehiculo y gasolina en cada viaje) .

S_i : De acuerdo al S se pondera con el indicador logistico, siendo $IL_i * S$ el costo por producto de acuerdo a su importancia.

$E(L_i)$: Los pedidos llegan al almacen en promedio 2 semanas después de realizar el pedido[1], por lo que el valor esperado del tiempo de entrega será 2/52.

σ_{L_i} : El proveedor puede adelantar o atrasar un pedido en promedio en 1 semana, por lo que se espera una desviación de 1/52.

CB_i : Dado el valor comercial y su relativo indicador logístico, se encuentra el costo por no tenerlo almacenado en el almacen.

tr_i : Dado el costo de utilización del vehiculo y gasolina en cada viaje se pondera con su respectivo indicador logístico.

Para los algoritmos se utilizó la estructura de datos mostrada en la Figura **5-1**. En la primera posición se almacena la Variable T, desde la posición 2 hasta el número de productos más dos, se almacenan las variables SS_i, donde i es el indice de los productos. Finalmente de la posición número de productos + 2 hasta la posición 2*número de productos+1 se almacena las variables k_i las cuales son enteras.

[1]Esto de acuerdo a lo dialogado con funcionarios de una empresa en salud prepagada

Tabla 5-1.: Valores y rangos de los parámetros utilizados en cada algoritmo uni-objetivo

AG	
Tamaño población	100
Probabilidad Mutación	[0.02, 0.2]
Selección	Rueda de ruleta
Cruce	Punto simple
PSO	
Particulas	100
Peso Inercial (ω)	[0.8, 1]
Aceleración Cognitiva (C_1)	[1.5, 2.5]
Aceleración Social (C_2)	[1.5, 2.5]
Factor de constricción (χ)	[0.03, 0.1]
CS	
Nidos	100
Probabilidad de abandono (P_a)	[0.2, 0.9]
BFOA	
Número de bacterias (S)	50
Número de iteraciones (Nc)	100
Número de veces en que la bacteria podrá nadar(Ns)	4
Número de reproducciones. (Nre)	4
Número de eliminaciones/disperciones (Ned)	2
Probabilidad de eliminación/disperción (Ped)	[0.15,0.4]

5.1. Aplicación de los algoritmos uni-objetivo propuestos

Por cada una de las cuatro funciones objetivo, se muestra el mejor desempeño histórico en las réplicas, el promedio y el desempeño en cada réplica de en cada algoritmo. Finalmente se muestra el vector de datos tal que minimice la función seleccionada, comparando el valor encontrado con las medidas de desempeño Tiempo Computacional, Mejor desempeño, Promedio del desempeño y Desviación del desempeño.

5.1.1. Función objetivo 1: Función de costo de inventario y pedido coordinado

En primer lugar se gráfico (Figura **5-2**) el mejor valor encontrado por cada uno de los algoritmos en las 100 réplicas, si en una réplica encontraba uno mejor que el histórico del algoritmo se modificaba como la mejor solución encontrada. Se observa que para la función objetivo 1 el de mejor desempeño fue el PSO, seguido por el BFOA, mientras que los algoritmos genéticos no obtiene un buen desempeño en esta función objetivo.

En la Figura **5-3** se observa que el BFOA en promedio siempre converge al mismo valor, mientras

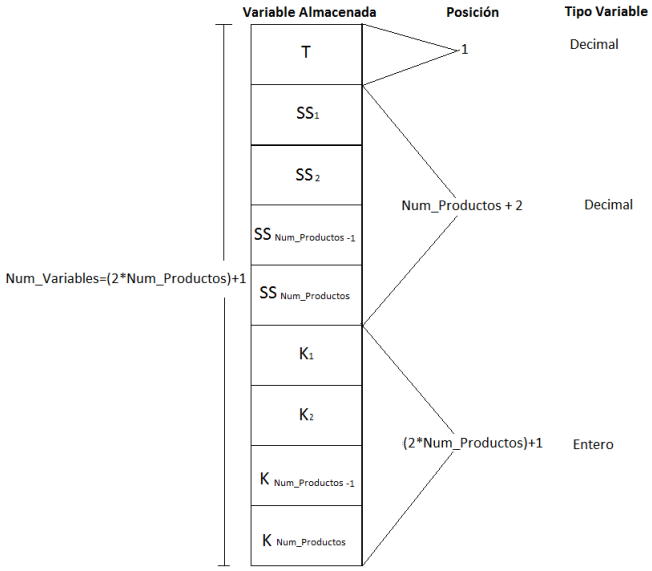

Figura 5-1.: Estructura del vector de los datos.

que el PSO presenta mayor variabilidad en los resultados obtenidos. El cuckoo search presenta una baja variabilidad pero un promedio alejado del mejor valor encontrado por el BFOA y PSO. El algoritmo genético presenta una alta variabilidad en los desempeños encontrados en cada réplica. En la Tabla mostrada en (Tabla **5-2**) se encuentra el vector de variables de la mejor solución encontrada por cada algoritmo, El PSO en las 100 réplicas encontró la mejor solución, aunque el BFOA obtuvo un mejor promedio, varianza y desviación en réplicas. El BFOA en tiempo computacional es el de menor desempeño mientras que el PSO tiene un tiempo razonable de ejecución.

De acuerdo a los resultados (Tabla **5-2**) el AG plantea que el tiempo entre reposiciones sucesivas es de 0,0085343 años, lo que equivale aproximadamente a 3 días. Al múltiplicarse este tiempo por el entero de coordinación nos da el tiempo de ciclo para cada producto i, por ejemplo, para el Fenobarbital el tiempo entre reposiciones será de 198 días con un factor de seguridad 0,97, mientras que el de Furosemida será de 120 días entre reposiciones con un factor de seguridad de 1,57.

5.1.2. Función objetivo 2: Función de frecuencia esperada de ocasiones de desabastecimiento anuales

En la Figura **5-4** se observa que los algoritmos AG, PSO y BFOA convergen hacia el mismo valor, mientras que el cuckoo search no converge. En la Figura **5-5** se observa que el de mayor variabilidad fue el Cuckoo search junto al PSO, mientras que en promedio el AG y BFOA siempre llegaron al mismo valor de desempeño.

En la Tabla **5-3** se muestra que AG y PSO encontraron el mismo valor de desempeño, pero el

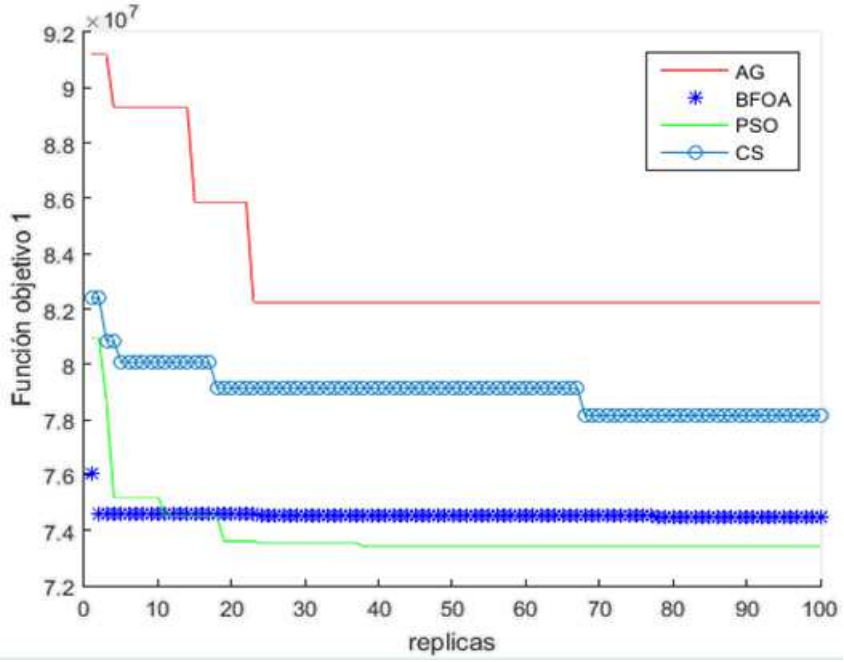

Figura 5-2.: Mejores desempeños históricos de cada uno de los algoritmos en la función de costo de inventario y pedido coordinado.

AG tiene menor variabilidad en cada réplica y tiene un menor tiempo computacional. El BFOA es el que menos varia entre réplicas pero tiene un alto gasto en tiempo computacional.

5.1.3. Función objetivo 3: Función de la cantidad esperada de productos faltantes anualmente

En la Figura **5-6** se observa que AG, PSO y BFOA convergen hacia el mismo valor, mientras que el cuckoo search no converge. En la Figura **5-7** el de mayor variabilidad fue el Cuckoo search y PSO, mientras que en promedio el AG y BFOA siempre llegaron al mismo valor de desempeño, se observa que tiene un comportamiento similar a la función de frecuencia esperada de ocasiones de desabastecimiento anuales.

En la Tabla **5-4** se muestra que AG y PSO encontraron el mismo valor de desempeño, pero el AG tiene menor variabilidad en cada réplica y tiene un menor tiempo computacional. El de menos variabilidad entre réplicas es BFOA pero tiene un alto gasto en tiempo computacional (en promedio 16 veces más que AG).

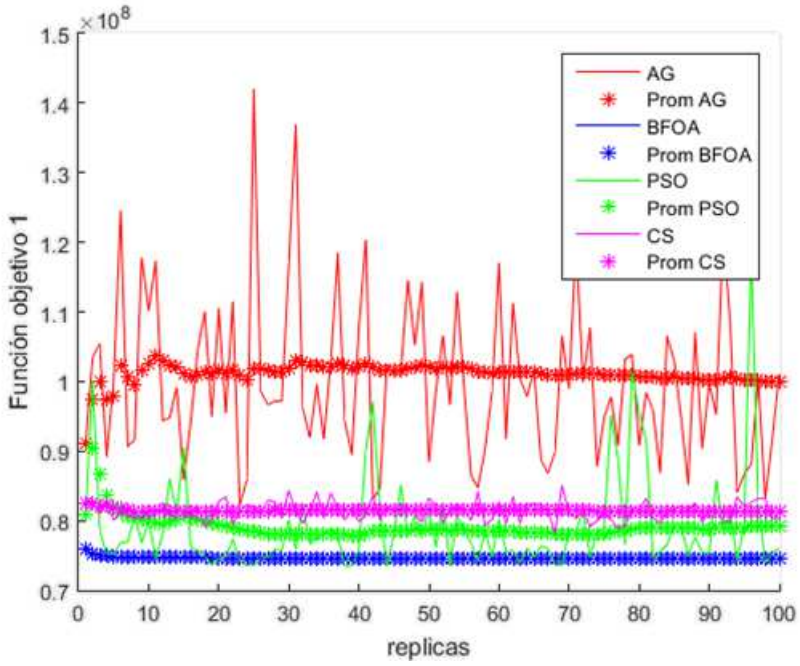

Figura 5-3.: Desempeño y promedio de los algoritmos en la función de costo de inventario y pedido coordinado.

5.1.4. Función objetivo 4: Función de costo de transporte del inventario

En la Figura **5-8** se observa que los algoritmos BFOA, CS y PSO convergen hacia el mismo valor, mientras que AG no converge. En la Figura **5-9** se observa que el de mayor variabilidad fue el AG junto al PSO, mientras que en promedio el CS y BFOA siempre llegaron al mismo valor de desempeño.

En la Tabla **5-5** se muestra que BFOA y PSO encontraron el mismo valor de desempeño, pero el PSO tiene menor variabilidad en cada réplica pero tiene un menor tiempo computacional. El BFOA es el que menos varia entre réplicas pero tiene un alto gasto en tiempo computacional (9 veces más en promedio que el PSO).

5.2. Comparación entre los desempeños obtenidos por algoritmos uni-objetivo

En la Tabla **5-6** se muestra un comparativo de los desempeños obtenidos por cada individuo seleccionado como el mejor en cada función objetivo utilizada contra el desempeño en las otras funciones.

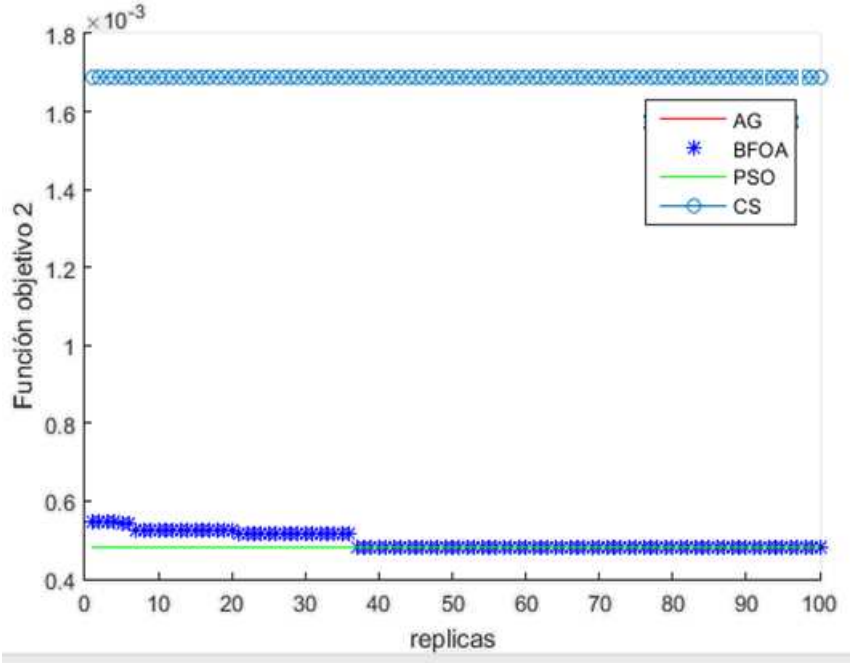

Figura 5-4.: Mejores desempeños históricos de cada uno de los algoritmos en la función de frecuencia esperada de ocasiones de desabastecimiento anuales.

Se observa que al minimizar la sólo Función de costo de inventario y pedido coordinado, se obtiene una frecuencia de pedidos atrasados de 14.5 y se espera quedar con faltantes de 14707 unidades. Si se minimiza sólo la Función de frecuencia esperada de ocasiones de desabastecimiento anuales o la Función de la cantidad esperada de productos faltantes anualmente se obtiene una menor cantidad de pedidos pendientes (atrasados) pero incrementa el costo notablemente en la Función de costo de inventario y pedido coordinado (pasa de 73 millones a 256 millones) y Función del costo de transporte (de 304 millones a 898 millones) siendo la función del costo de transporte la más afectada. Si sólo se minimiza la función del costo de transporte, aumenta la frecuencia de pedidos atrasados y unidades faltantes, pero se minimiza el costo de la Función de costo de inventario y pedido coordinado. Dado lo anterior se comprueba la contradicciones entre los objetivos de costo de inventario, pedido coordinado y costo de transporte contra la frecuencia esperada de ocasiones de desabastecimiento anuales y cantidad esperada de productos faltantes anualmente, por lo que se aplicaran algoritmos multi-objetivo.

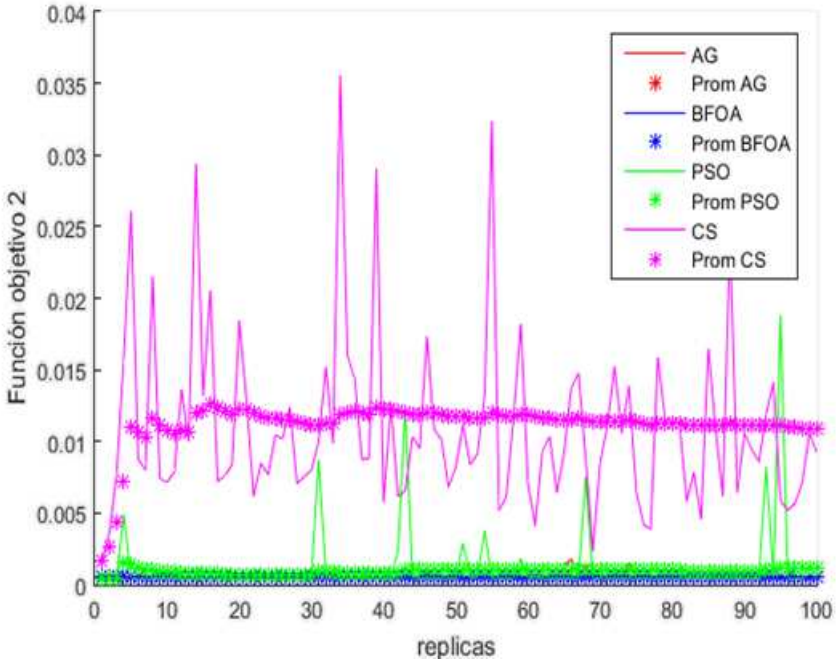

Figura 5-5.: Desempeño y promedio de los algoritmos en la función de frecuencia esperada de ocasiones de desabastecimiento anuales.

5.3. Aplicación de algoritmos bio-inspirados multi-objetivo

El problema multi-objetivo propuesto se solucionó mediante algoritmos bio-inspirados multi-objetivos tales como NSGA II, NSPSO, MOCS y BCMOA para obtener la frontera óptima de Pareto. Las soluciones encontradas se comparan mediante métricas como el indicador de dominancia de Pareto, diversidad y tiempo computacional; obteniendo una frontera de Pareto con aportes de todos los algoritmos. Finalmente se aplicó TOPSIS seleccionando el punto más cercano a la solución ideal.

En la Figura **5-10** se observa la conformación de las fronteras de Pareto de cada uno de los algoritmos, en está Figura se muestran diferentes vistas en el espacio de búsqueda. Se observa que los Algorimos BCMOA, NSGA II y NSPSO forman fronteras similares entre ellos, mientras que MOCS no forma una frontera definida.

Al observar la frontera de Pareto resultante en la Figura **5-11** se aprecia que lo aportan los algoritmos BCMOA y NSGA II. Al aplicar TOPSIS sobre los datos no dominados de todos los algoritmos se encontro un equilibrio entre los costos de inventario, coordinación y transporte con la cantidad y frecuencia esperada de unidades faltantes. El punto TOPSIS, encontrado refleja un comportamiento interesante en el que la mejor solución se ubica en el máximo valor posible para la función del de costo de inventario y pedido coordinado (Función 1) mientras que al mismo tiempo se localiza en el minimo asumible para los demÃ¡s objetivos, haciendo que la mejor solución sea alcanzable a partir del sacrificio del los costo de inventario y pedido coordinado

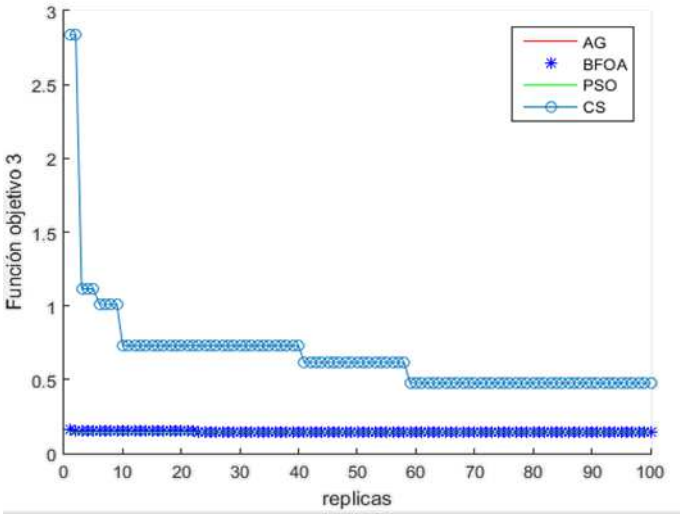

Figura 5-6.: Mejores desempeños históricos de cada uno de los algoritmos en la Función de la cantidad esperada de productos faltantes anualmente.

En la Tabla **5-8** se muestran las medidas de desempeño de los algoritmos, en cuánto al **tiempo computacional** el NSPSO Y BCMOA obtienen los mejores desempeños. El NSGA II es el que presenta mayor **Diversidad** seguido de NSPSO. En la frontera resultante se tiene que el algoritmo que más **dominancia de Pareto** obtuvo fue el BCMOA, 51 porciento, mientras que el NSGA II aportó el 48 porciento. El MOCS no muestra un buen desempeño para este tipo de problema de inventario, dado los pocos puntos que convergen a la frontera. Entre los algoritmos desarrollados el BCMOA y NSGA II fueron los de mejor desempeño dado su aporte a la conformación de la frotera de Pareto, en tiempos aceptables de solución y obteniendo una diversidad buena en las soluciones encontradas.

En la Tabla **5-10** se muestra que el múltiplo entero del ciclo de reposiciones sucesivas de tiempo para todos los productos fue de $k_i = 40 \forall i$, el tiempo entre pedidos de manera coordinada es $T_i = k_i * T = 40 * 0,0085 = 0,34$ años (17.6 semanas aproximadamente) para todos los productos, dado que para todos los productos, el factor de seguridad para cada producto es cercano a 3,9 el cual indica el aumento de unidades en el punto de reposición con el fin de disminuir los pendientes (faltantes). La Tabla **5-11** se calcula el $Q_i = T_i * E(D_i)$ y $R_i = SS_i \sigma_{DLi} + \mu_{DLi}$, encontrando asi las cantidades de pedido y punto de reposición respectivamente, por ejemplo de Metformina se debe pedir 4788 unidades y se debe ordenar cuando el nivel de inventario este por debajo de 3195 unidades.

De igual manera se observa en la Tabla **5-9** el costo de inventario y pedido coordinado (135 millones), frecuencia esperada de ocasiones de desabastecimiento anuales (0.0014 ocasiones), cantidad esperada de faltantes (1.38 unidades) y el costo de transporte del inventario (305 millones), es una solución óptima a comparación de las mostradas en la Tabla **5-6**.

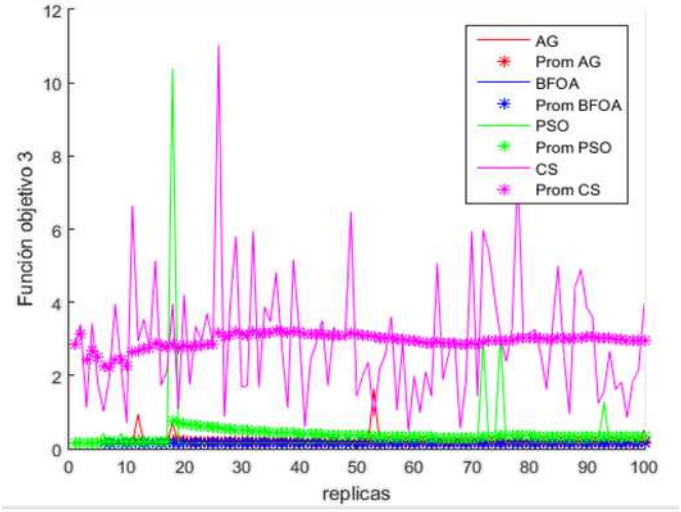

Figura 5-7.: Desempeño y promedio de los algoritmos en la función de la cantidad esperada de productos faltantes anualmente.

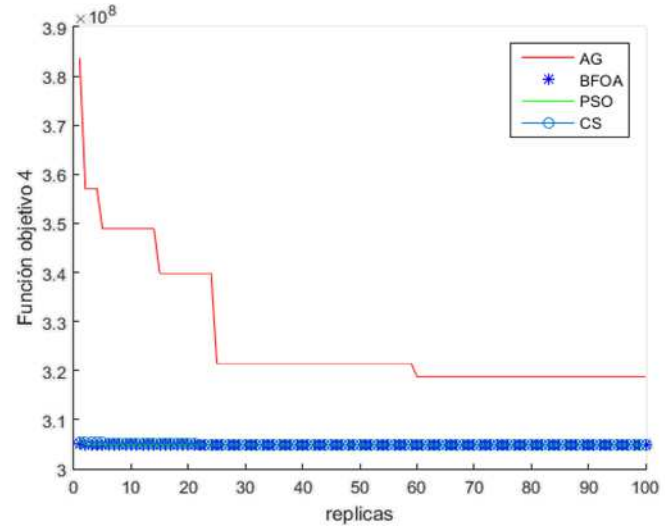

Figura 5-8.: Mejores desempeños históricos de cada uno de los algoritmos en la función de costo de transporte del inventario.

Tabla 5-2.: Tabla resumen de las medidas de desempeño de la función costo de inventario y pedido coordinado.

		Función Objetivo 1			
		AG	**PSO**	**CS**	**BFOA**
Tiempo entre reposiciones sucesivas (años) T		0,0085343	0,0085343	0,0085343	0,0085343
Factor de seguridad del producto i (SS)	Fenobarbital	0,93764922	0,02393167	1,52901503	0,2208266
	Eritromicina comp.	0,32620654	0,00399064	0,77961574	0,1247908
	Metformina	0,33172993	0,01053929	0,25999379	0,00838739
	Amoxicilina comp	0,54194204	0,00339644	0,05662711	0,04965922
	Furosemida	1,57331309	0,01748199	1,68111423	0,09582271
	Glibenclamida	1,24335505	0,00668637	1,71536123	0,63835984
	Hidroclorotiazida	0,20885001	0,00293715	0,18572384	0,01714474
	Ácido Acetilsalicílico	0,06520156	0,00070367	0,10233189	0,01543374
	Nitrendipino	2,06791873	0,01788838	1,9727642	0,40616478
	Enalapril	2,39331939	0,00789003	1,0295568	0,63650297
Entero coordinación(k_i)	Fenobarbital	66	40	40	40
	Eritromicina comp.	48	40	40	40
	Metformina	53	40	40	40
	Amoxicilina comp	46	40	40	40
	Furosemida	40	40	40	40
	Glibenclamida	53	40	40	40
	Hidroclorotiazida	40	40	40	40
	Ácido Acetilsalicílico	40	40	40	40
	Nitrendipino	43	40	40	40
	Enalapril	42	40	40	40
Mejor Fitnes encontrado en réplicas		82250377,2	73411923,5	78160999	74480224,7
Promedio réplicas		99870431,3	79193689,9	81374194,3	74563124,5
Varianza réplicas		1,3349E+14	4,9746E+13	1,9079E+12	2,4414E+10
Desviación réplicas		11553725,6	7053096,6	1381252,49	156248,32
Tiempo computacional (Segundos)		168	525	1096	5871

Tabla 5-3.: Tabla resumen de las medidas de desempeño de la función de frecuencia esperada de ocasiones de desabastecimiento anuales.

		Función Objetivo 2			
		AG	**PSO**	**CS**	**BFOA**
Tiempo entre reposiciones sucesivas (años) T		0,025	0,02	0,025	0,025
Factor de seguridad del producto i (SS)	Fenobarbital	3,9	3,9	3,27	3,9
	Eritromicina comp.	3,9	3,9	3,9	3,9
	Metformina	3,9	3,9	3,75	3,9
	Amoxicilina comp	3,9	3,9	3,58	3,9
	Furosemida	3,9	3,9	3,87	3,9
	Glibenclamida	3,9	3,9	3,64	3,89
	Hidroclorotiazida	3,9	3,9	3,45	3,9
	Ácido Acetilsalicílico	3,9	3,9	3,46	3,9
	Nitrendipino	3,9	3,9	3,77	3,9
	Enalapril	3,9	3,9	3,88	3,9
Entero coordinación(k_i)	Fenobarbital	40	50	40	40
	Eritromicina comp.	40	50	40	40
	Metformina	40	50	40	40
	Amoxicilina comp	40	50	40	40
	Furosemida	40	50	40	40
	Glibenclamida	40	50	40	40
	Hidroclorotiazida	40	50	40	40
	Ácido Acetilsalicílico	40	50	40	40
	Nitrendipino	40	50	40	40
	Enalapril	40	50	40	40
Mejor Fitnes Encontrado en réplicas		0,00048096	0,00048096	0,00169	0,0004825
Promedio réplicas		0,00055	0,0011	0,0109	0,00049
Varianza réplicas		4,58E-8	6,54E-6	3,76E-5	4,72E-10
Desviación réplicas		0,00021	0,0025	0,0061	2,17E-5
Tiempo computacional (Segundos)		824	1200	1804	15149

Tabla 5-4.: Tabla resumen de las medidas de desempeño de la función de la cantidad esperada de productos faltantes anualmente.

		Función Objetivo 3			
		AG	**PSO**	**CS**	**BFOA**
Tiempo entre reposiciones sucesivas (años) T		0,025	0,015	0,025	0,025
Factor de seguridad del producto i (SS)	Fenobarbital	3,9	3,9	3,9	3,9
	Eritromicina comp.	3,9	3,9	3,2	3,79
	Metformina	3,9	3,9	3,9	3,84
	Amoxicilina comp	3,9	3,9	3,46	3,9
	Furosemida	3,9	3,9	3,64	3,9
	Glibenclamida	3,9	3,9	3,58	3,9
	Hidroclorotiazida	3,9	3,9	3,67	3,9
	Ácido Acetilsalicílico	3,9	3,9	3,3	3,9
	Nitrendipino	3,9	3,9	3,84	3,9
	Enalapril	3,9	3,9	3,89	3,89
Entero coordinación(k_i)	Fenobarbital	40	63	40	40
	Eritromicina comp.	40	63	40	40
	Metformina	40	63	40	40
	Amoxicilina comp	40	63	40	40
	Furosemida	40	63	40	40
	Glibenclamida	40	63	40	40
	Hidroclorotiazida	40	63	40	40
	Ácido Acetilsalicílico	40	63	40	40
	Nitrendipino	40	63	40	40
	Enalapril	40	63	40	40
Mejor Fitnes Encontrado en réplicas		0,1409	0,1409	0,47	0,145
Promedio réplicas		0,18	0,32	2,95	0,14
Varianza réplicas		0,03	1,2	2,93	1,31E-5
Desviación réplicas		0,18	1,098	1,71	0,0036
Tiempo computacional (Segundos)		926	1414	1729	15952

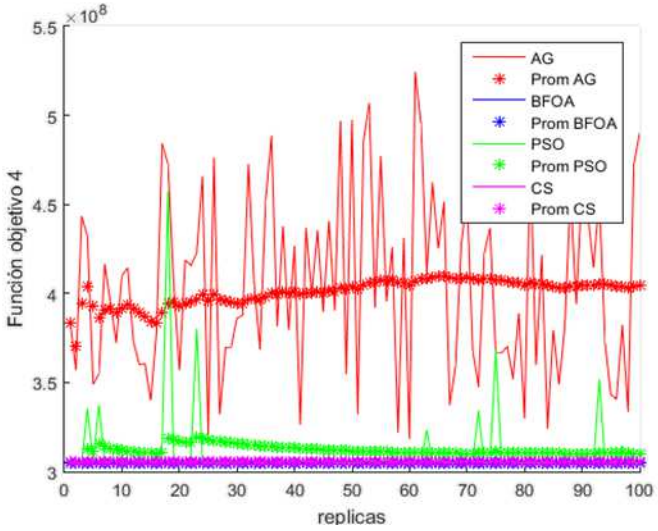

Figura 5-9.: Desempeño y promedio de los algoritmos en la función de costo de transporte del inventario.

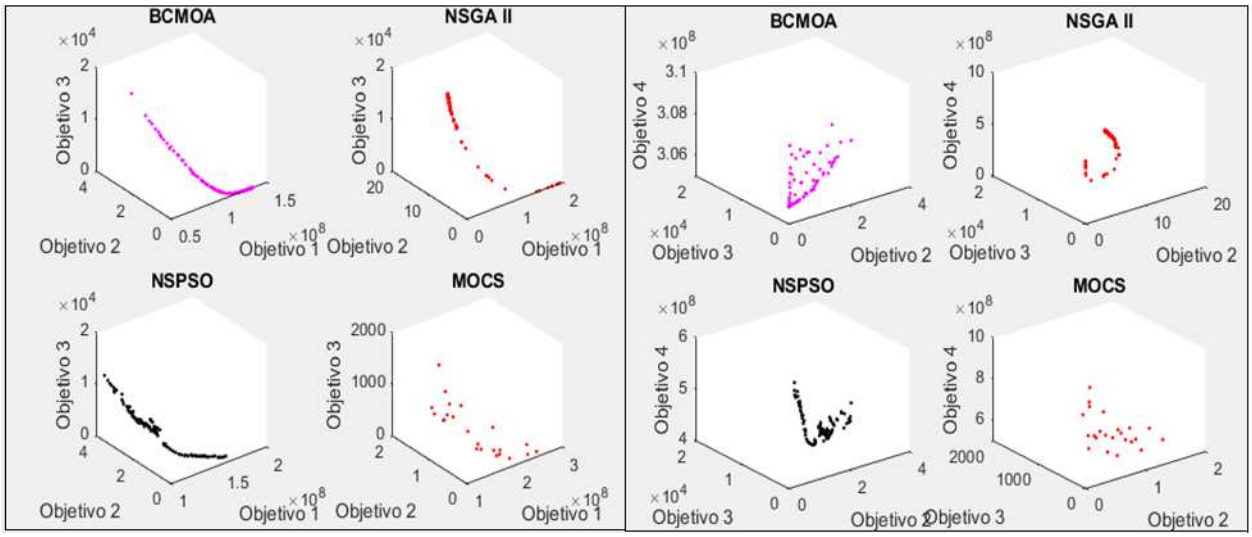

Figura 5-10.: Fronteras de Pareto obtenidas por cada uno de los algoritmos desde diferentes vistas.

Tabla 5-5.: Tabla resumen de las medidas de desempeño de la función costo de transporte del inventario.

		Función Objetivo 4			
		AG	**PSO**	**CS**	**BFOA**
Tiempo entre reposiciones sucesivas (años) T		0,0085343	0,0085343	0,0085343	0,0085343
Factor de seguridad del producto i (SS)	Fenobarbital	3,32	2,96	3,54	3,64
	Eritromicina comp.	0,95	3,9	1,75	1,61
	Metformina	3,5	1,61	1,091	0,95
	Amoxicilina comp	3,31	0,046	3,39	3,50
	Furosemida	1,76	3,34	3,22	2,39
	Glibenclamida	1,84	0,71	1,89	3,15
	Hidroclorotiazida	3,9	3,9	2,39	0,36
	Ácido Acetilsalicílico	2,78	0,0067	1,22	1,52
	Nitrendipino	1,32	0,051	3,48	2,05
	Enalapril	3,29	2,63	3,9	3,44
Entero coordinación(k_i)	Fenobarbital	46	40	41	40
	Eritromicina comp.	42	40	40	40
	Metformina	40	40	40	40
	Amoxicilina comp	43	40	40	40
	Furosemida	44	40	40	40
	Glibenclamida	43	40	40	40
	Hidroclorotiazida	43	40	40	40
	Ácido Acetilsalicílico	40	40	40	40
	Nitrendipino	40	40	40	40
	Enalapril	44	40	40	40
Mejor Fitnes Encontrado en réplicas		318788281	304923033	304970697	304923033
Promedio réplicas		404421990	310098574	305480614	304926483
Varianza réplicas		2,64E+15	3,62E+14	1,31E+10	1,64E+8
Desviación réplicas		51364922,9	19021112,5	114508,879	12815,7
Tiempo computacional (Segundos)		107	443	450	4096

Tabla 5-6.: Comparación entre los desempeños obtenidos por algoritmos uni-objetivo

		Valores tomados en las otras funciones			
		F1	**F2**	**F3**	**F4**
Función	**F1**	$ 73.411.923	14,5	14707,0	$ 305.809.165
	F2	$ 256.478.545	0,00048096	0,14052588	$ 898.426.463
	F3	$ 256.478.545	0,00047957	0,14093418	$ 898.426.463
	F4	$ 95.623.255	1,97112585	1333,12698	$ 304.923.033

Tabla 5-7.: Valores y rangos de los parámetros utilizados en cada algoritmo Multi-objetivo

NSGA II	
Tamaño población	100
Número de generaciones	100
pX probabilidad de cruce	1
pM probabilidad de mutación	0.1
NSPSO	
Número de iteraciones	100
Particulas	100
Peso Inercial (ω)	[0.4, 1]
Aceleración Cognitiva (C_1)	2
Aceleración Social (C_2)	2
Factor de constricción (χ)	0.03
MOCS	
Número de vuelos	100
Nidos	500
Probabilidad de abandono (P_a)	0.9
BCMOA	
Número de bacterias (S)	100
Número de iteraciones (Nc)	100

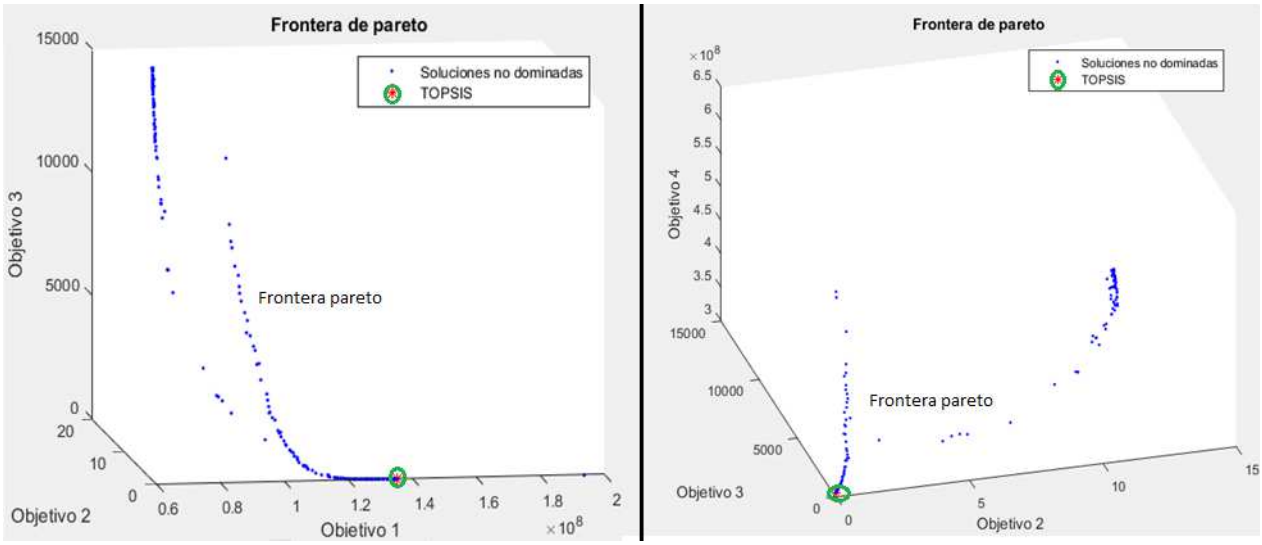

Figura 5-11.: Frontera de Pareto resultante con la selección de la alternativa ideal mediante TOPSIS.

Tabla 5-8.: Medidas de desempeño para los algoritmos multi-objetivo usados

	NSGA II	**NSPSO**	**BCMOA**	**MOCS**
Tiempo Computacional	103	60	62	121
Diversidad	1,5918	0,7949	0,585	0,5331
Dominancia Pareto	0,4845	0	0,5155	0

Tabla 5-9.: Desempeño de las funciones objetivo evaluadas con la solución encontrada por TOPSIS

F1	**F2**	**F3**	**F4**
$135.441.626,51	0,00144207	1,38935964	$l 305.814.630,61

Tabla 5-10.: Mejor alternativa seleccionada de la frontera de Pareto mediante TOPSIS

Mejor alternativa seleccionada por TOPSIS		
Tiempo entre reposiciones sucesivas (años) T		0,0085
Factor de seguridad del producto SS_i	Fenobarbital	3,8997
	Eritromicina comp.	3,8961
	Metformina	3,8929
	Amoxicilina comp	3,8831
	Furosemida	3,9000
	Glibenclamida	3,8981
	Hidroclorotiazida	3,8877
	ÁAcido Acetilsalicílico	3,8924
	Nitrendipino	3,8997
	Enalapril	3,8946
Múltiplo entero del ciclo k_i	Fenobarbital	40
	Eritromicina comp.	40
	Metformina	40
	Amoxicilina comp	40
	Furosemida	40
	Glibenclamida	40
	Hidroclorotiazida	40
	ÁAcido Acetilsalicílico	40
	Nitrendipino	40
	Enalapril	40

Tabla 5-11.: Cantidad a pedir y punto de reposición para cada uno de los productos de acuerdo a la alternativa seleccionada

Producto	Q_i	R_i
Fenobarbital	1246	493
Eritromicina comp.	1646	672
Metformina	4788	3195
Amoxicilina comp	5198	2331
Furosemida	5701	3766
Glibenclamida	9471	5599
Hidroclorotiazida	9538	6288
Ácido Acetilsalicílico	14421	8869
Nitrendipino	15756	9998
Enalapril	32883	20122

6. Simulación de diferentes escenarios para la validación de los algoritmos

6.1. Generación de parámetros aleatorios

Se generaron escenarios para 5, 10, 20, 35 y 50 productos de acuerdo a los parámetros aleatorios uniformemente distribuidos según se observa en la Tabla 6.1. Esto con el fin de ver el desempeño de los algoritmos propuestos en diferentes escenarios hasta con 50 productos para ser coordinados.

Tabla 6-1.: Valores y rangos de cada uno de los parámetros asociados

Parámetros	**Valores o rango**
h_i	UNIF [0.2, 3]
$E(D_i)$	UNIF [100, 100000]
σ_{Di}	$\frac{E(D_i)}{300}$
S	UNIF [5, 20]
S_i	UNIF [0.5, 5]
$E(L_i)$	UNIF[0.5, 24]
σ_{L_i}	$\frac{E(L_i)}{30}$
CB_i	UNIF [5, 14]
tr_i	UNIF [0.5, 5]

6.2. Aplicación de los algoritmos uni-objetivo

Se aplicaron los algoritmos uni-objetivos propuestos con el fin de observar el mejor desempeño independientemente en cada una de las funciones objetivo propuestas. El BFOA soló se aplicó para 5 y 10 productos debido a su alto costo en tiempo computacional.

En las Figuras **6-1**, **6-2**, **6-3** y **6-4** se observa el desempeño de los algoritmos en cada una de las funciones objetivo. En las Funciones 1 y 4 para 5 y 10 productos el PSO y CS convergen casi al mismo valor minimo en todas las réplicas, en las Funciones 2 y 3 para 5 y 10 productos el PSO y AG convergen casi al mismo valor minimo en todas las réplicas. Se observa que para 20, 35 y 50 productos el PSO es el de mejor desempeño, mientras que los AG y CS se alejan del mejor valor encontrado a medida que aumenta el número de productos a ser coordinados.

Tabla 6-2.: Tabla resumen de las medidas de desempeño de la función 1.

		F1			
		AG	PSO	CS	BFOA
5 Productos	Mejor Fitness encontrado en réplicas	10802	4459	3632	26403
	Promedio réplicas	43627	8955	5068	35605
	Varianza réplicas	223014042	76092344	389416	34041164
	Desviación réplicas	14934	8723	624	5834
	Tiempo computacional (Segundos)	221	395	660	3964
10 Productos	Mejor Fitness encontrado en réplicas	277304,195	10988,4028	60539,4649	282548,82
	Promedio réplicas	534095,246	93786,6433	117793,042	339088,733
	Varianza réplicas	1,3273E+10	1,0769E+10	1238766555	1303050819
	Desviación réplicas	115209,675	103772,943	35196,1156	36097,7952
	Tiempo computacional (Segundos)	357	705	725	6435
20 Productos	Mejor Fitness encontrado en réplicas	1325109,502	20883,87821	824655,945	N.A.
	Promedio réplicas	2027947,42	539233,9674	1513142,1	N.A.
	Varianza réplicas	92516716409	1,53341E+11	7,9133E+10	N.A.
	Desviación réplicas	304165,6069	391588,4104	281306,603	N.A.
	Tiempo computacional (Segundos)	629	1239	1507	N.A.
35 Productos	Mejor Fitness encontrado en réplicas	1236036,38	93155,9982	1310097,97	N.A.
	Promedio réplicas	1872796,56	753238,157	1661866,94	N.A.
	Varianza réplicas	4,4747E+10	1,1659E+11	2,978E+10	N.A.
	Desviación réplicas	211535,651	341452,532	172567,688	N.A.
	Tiempo computacional (Segundos)	1041	2149	2505	N.A.
50 Productos	Mejor Fitness encontrado en réplicas	1891170,327	414563,0684	2117483,98	N.A.
	Promedio réplicas	3013092,225	1371866,452	2684627,74	N.A.
	Varianza réplicas	1,07951E+11	3,03753E+11	3,8881E+10	N.A.
	Desviación réplicas	328559,1946	551138,1177	197182,343	N.A.
	Tiempo computacional (Segundos)	1562	3378	5050	N.A.

En las Figuras **6-5**, **6-6**, **6-7** y **6-8** se observa que para las Funciones 1 y 4 en todos los productos el PSO presenta una alta variabilidad, junto a los AG mientras que el CS y BFOA son más estables. Para las Funciones 2 y 3 el AG presenta una alta variabilidad en las respuestas obtenidas en cada réplica, mientras que el PSO para estas funciones es más estable.

En las Tablas **6-2**, **6-3**, **6-4** y **6-5** se observa que el PSO fue el de mejor desempeño pero presento más promedio, varianza y desviación en las réplicas. Aunque en tiempo computacional es segundo solo siendo superado por los AG. Se puede concluir que si se quiere optimizar una sola función objetivo el algoritmo recomendado será PSO aunque se deben hacer un número considerable de réplicas para evitar tener óptimos locales.

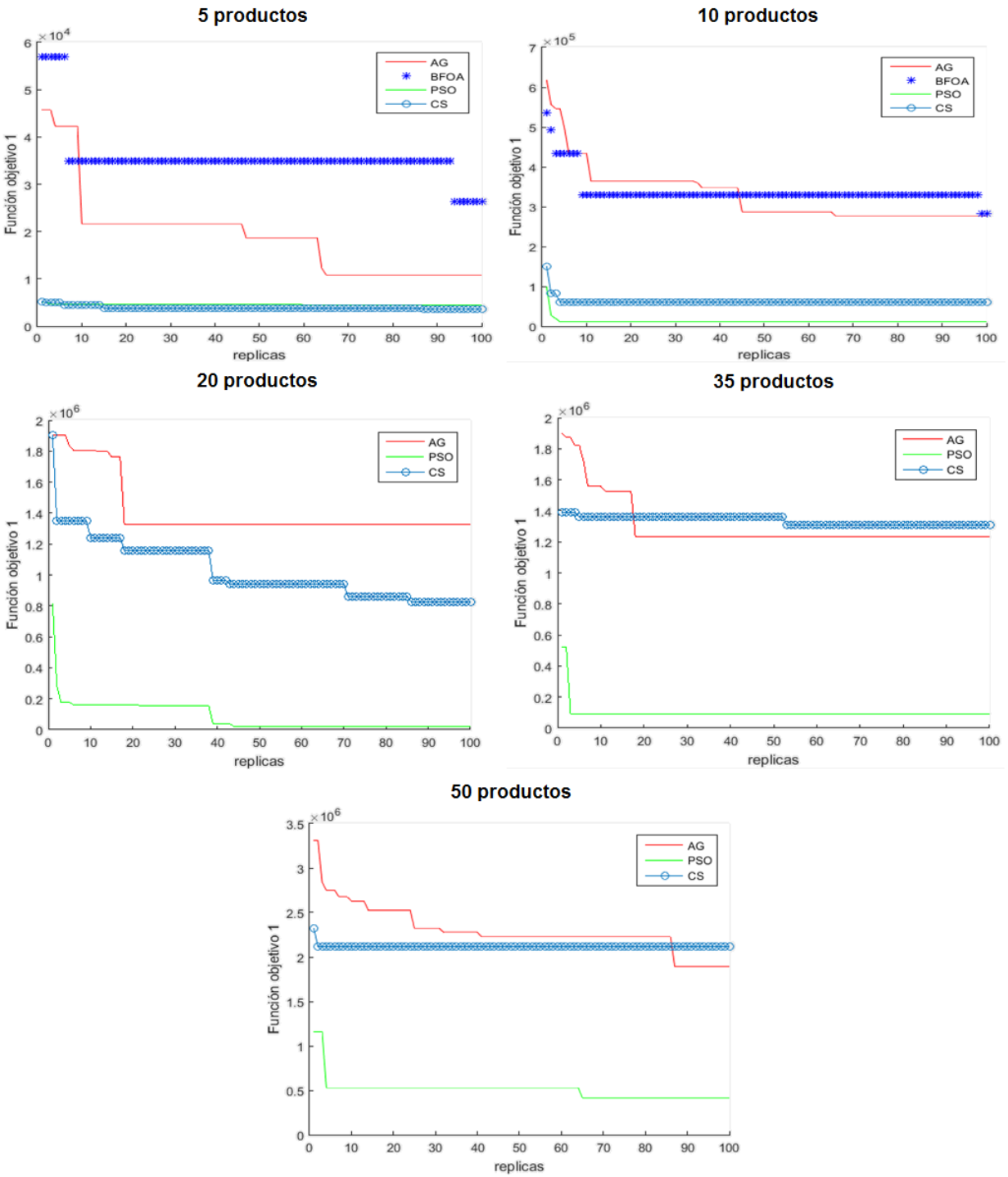

Figura 6-1.: Mejores desempeños históricos de cada uno de los algoritmos en la función 1.

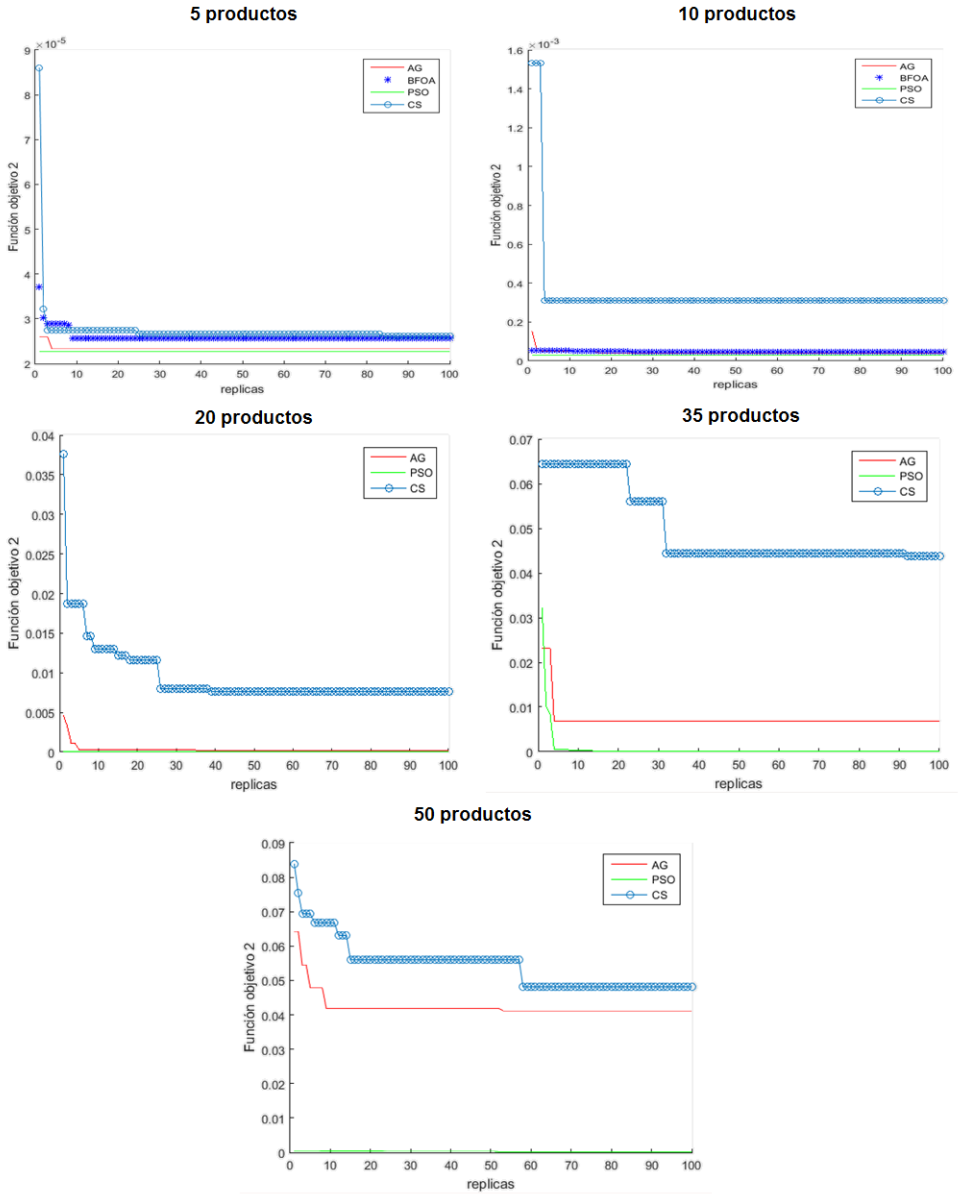

Figura 6-2.: Mejores desempeños históricos de cada uno de los algoritmos en la función 2.

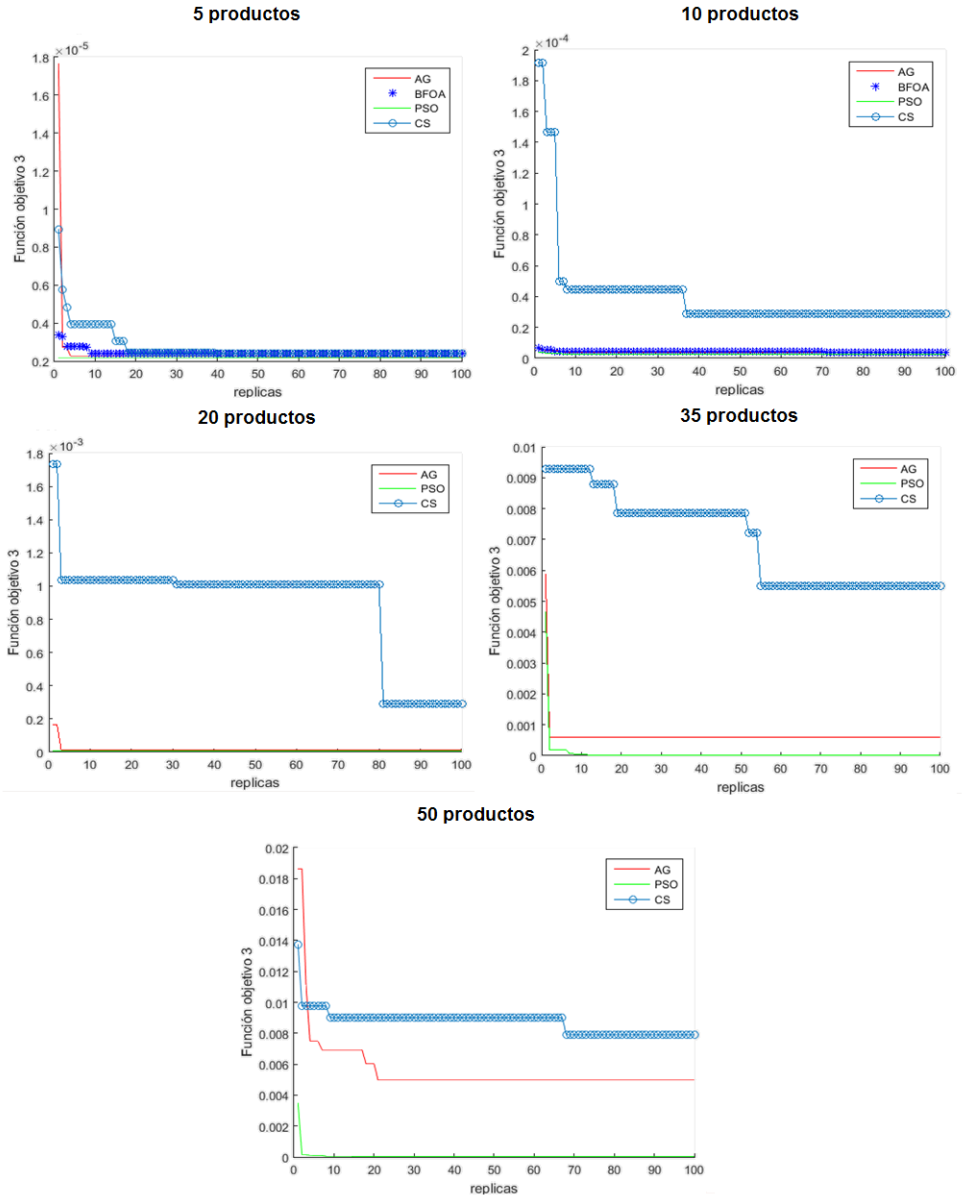

Figura 6-3.: Mejores desempeños históricos de cada uno de los algoritmos en la función 3.

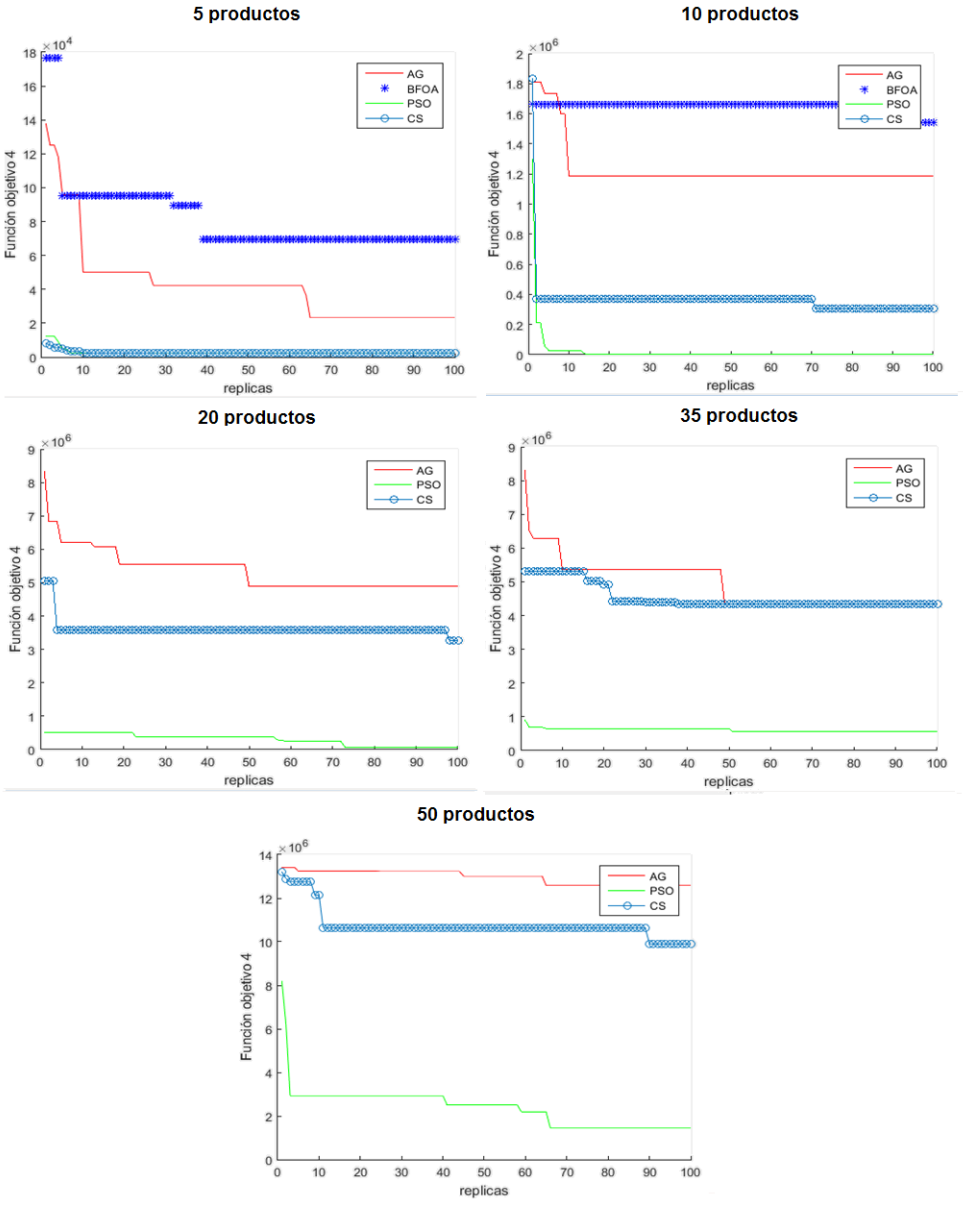

Figura 6-4.: Mejores desempeños históricos de cada uno de los algoritmos en la función 4.

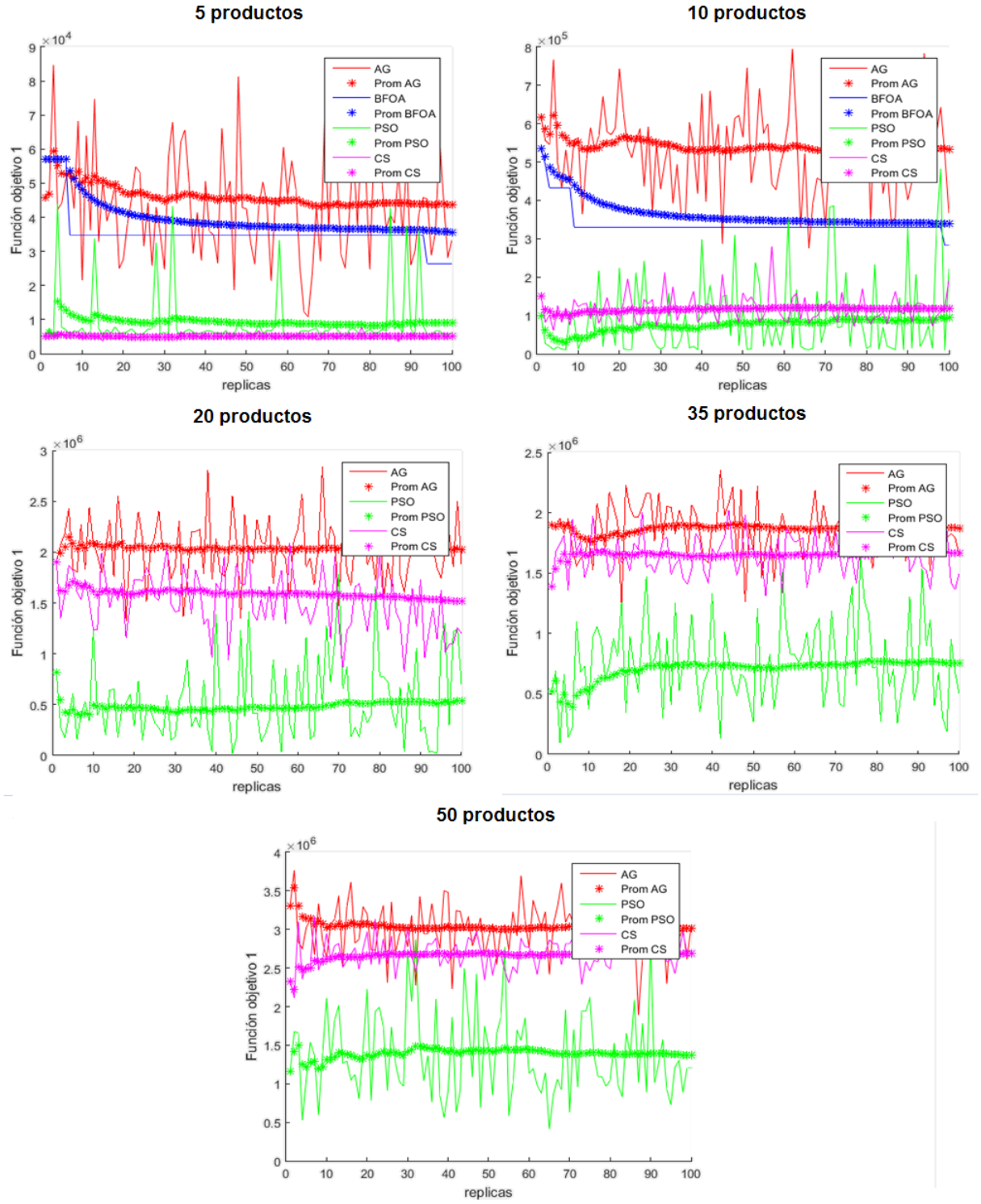

Figura 6-5.: Desempeño y promedio de los algoritmos en la función 1.

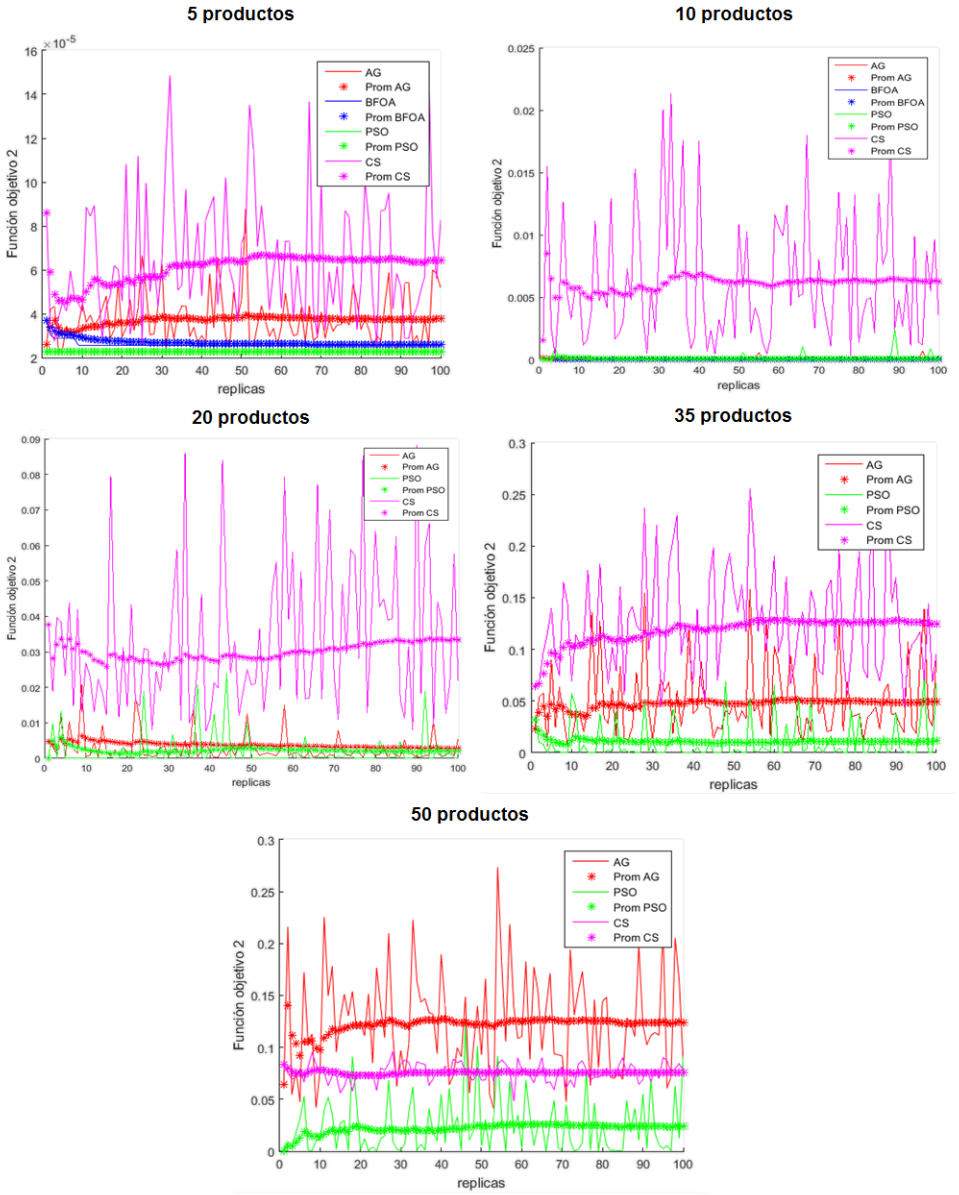

Figura 6-6.: Desempeño y promedio de los algoritmos en la función 2.

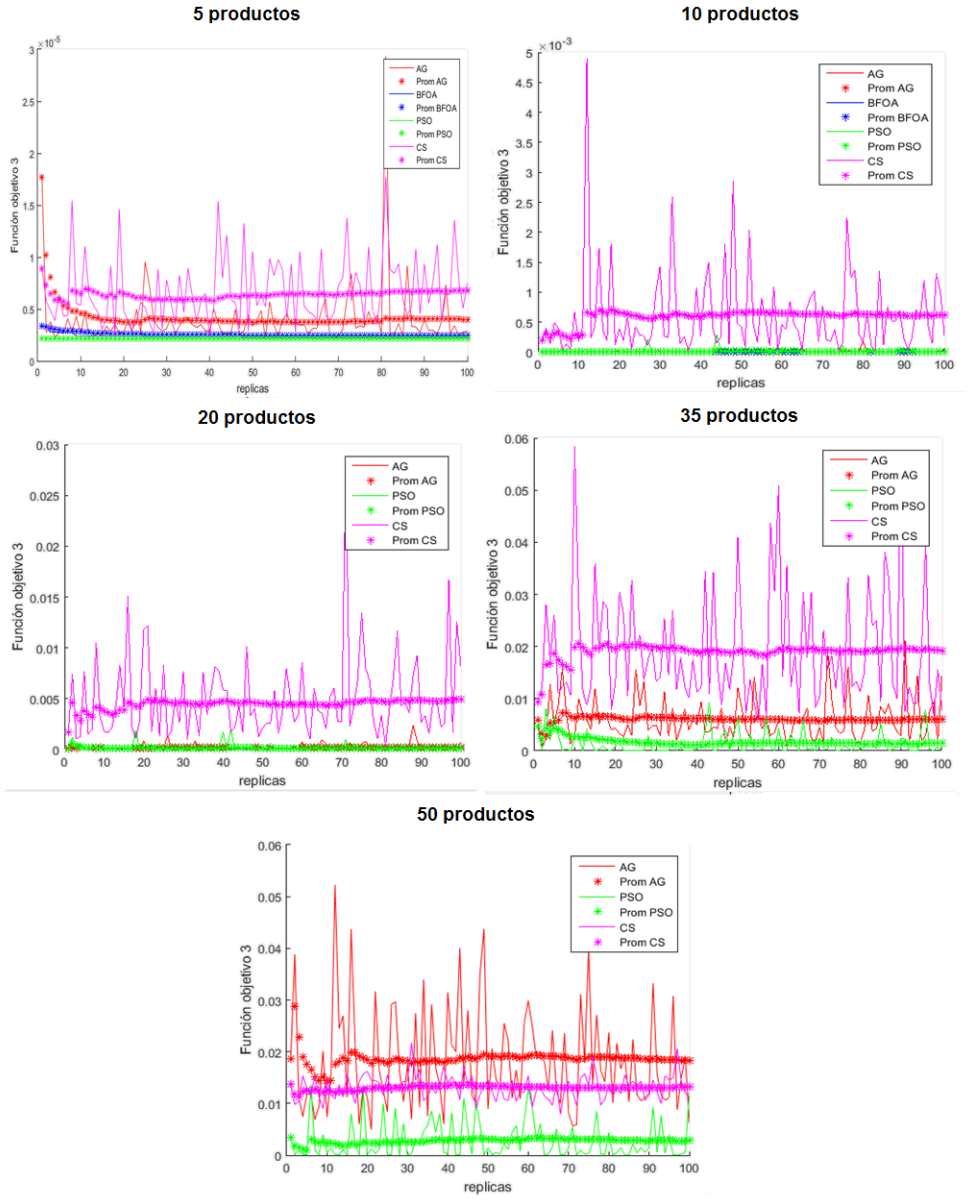

Figura 6-7.: Desempeño y promedio de los algoritmos en la función 3.

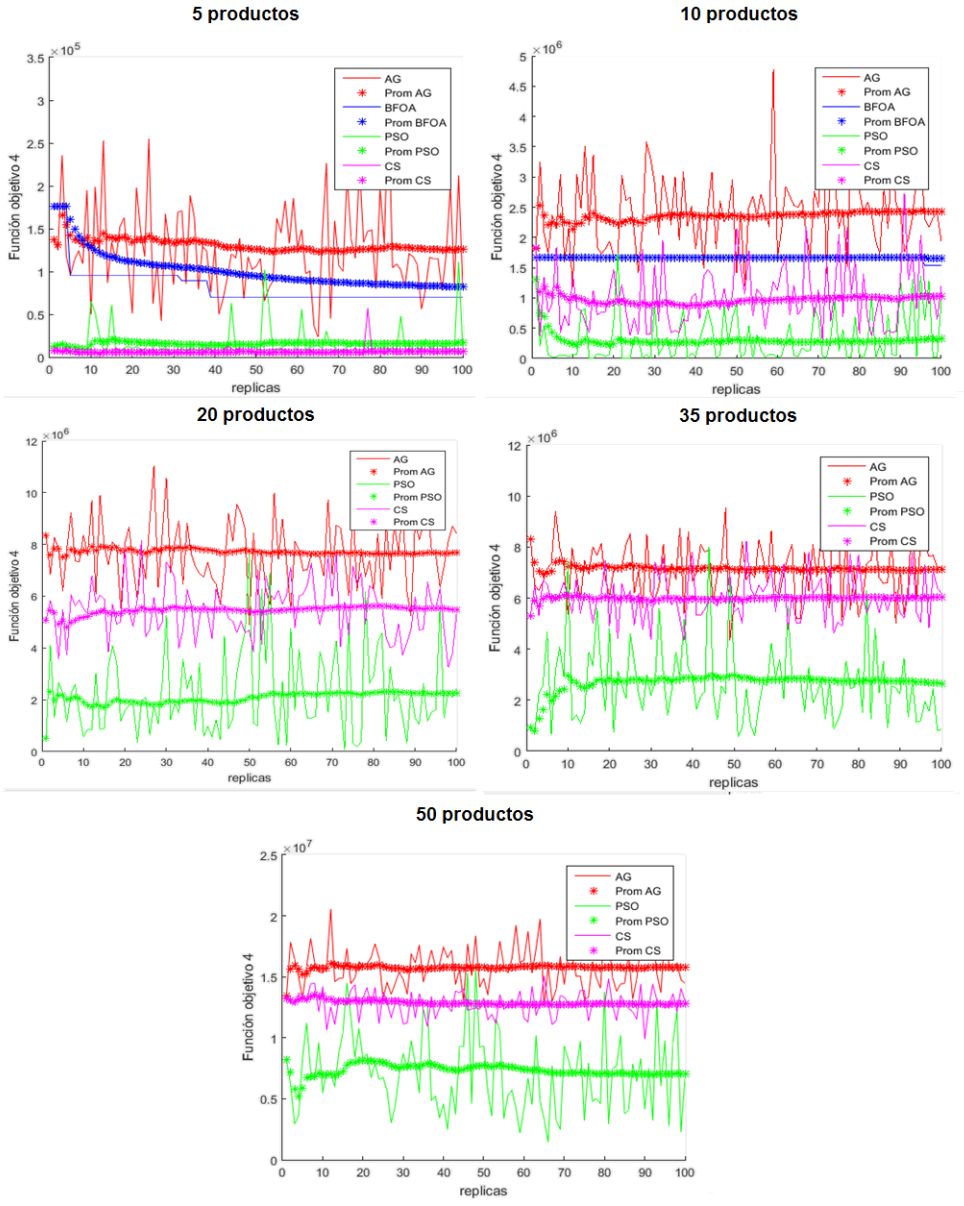

Figura 6-8.: Desempeño y promedio de los algoritmos en la función 4.

Tabla 6-3.: Tabla resumen de las medidas de desempeño de la función 2.

		F2			
		AG	PSO	CS	BFOA
5 Productos	Mejor Fitness encontrado en réplicas	2,33E-05	2,27E-05	2,61E-05	2,56E-05
	Promedio réplicas	3,78E-05	2,27E-05	6,44E-05	2,59E-05
	Varianza réplicas	1,34E-06	9,39E-40	7,87E-10	2,06E-12
	Desviación réplicas	0,001158922	3,06468E-20	2,8057E-05	1,4E-06
	Tiempo computacional (Segundos)	626	712	906	7879
10 Productos	Mejor Fitness encontrado en réplicas	4,09E-05	3,12E-05	0,00030982	4,52E-05
	Promedio réplicas	0,00010126	0,00010106	0,00628165	4,69E-05
	Varianza réplicas	7,29E-09	8,24E-08	2,48E-05	6,99E-12
	Desviación réplicas	8,54E-05	0,000287	0,00498184	2,64E-06
	Tiempo computacional (Segundos)	1098	1468	1572	16047
20 Productos	Mejor Fitness encontrado en réplicas	0,000231693	7,06E-05	0,00767968	N.A.
	Promedio réplicas	0,002851813	0,001963598	0,03340665	N.A.
	Varianza réplicas	1,52E-05	2,10E-05	0,00043116	N.A.
	Desviación réplicas	0,003904087	0,004583904	0,02076449	N.A.
	Tiempo computacional (Segundos)	2027	2784	2999	N.A.
35 Productos	Mejor Fitness encontrado en réplicas	0,0067825	0,0001622	0,0436854	N.A.
	Promedio réplicas	0,04976384	0,01152556	0,12476763	N.A.
	Varianza réplicas	0,00121624	0,00036396	0,00261434	N.A.
	Desviación réplicas	0,03487458	0,01907783	0,05113058	N.A.
	Tiempo computacional (Segundos)	3458	4612	5479	N.A.
50 Productos	Mejor Fitness encontrado en réplicas	0,041190902	0,00017878	0,04811398	N.A.
	Promedio réplicas	0,124006505	0,024268218	0,07580441	N.A.
	Varianza réplicas	0,002359345	0,000741253	8,25E-05	N.A.
	Desviación réplicas	0,048573092	0,02722596	0,0090811	N.A.
	Tiempo computacional (Segundos)	4734	6440	8296	N.A.

Tabla 6-4.: Tabla resumen de las medidas de desempeño de la función 3.

		F3			
		AG	PSO	CS	BFOA
5 Productos	Mejor Fitness encontrado en réplicas	2,21E-06	2,18E-06	2,41E-06	2,42E-06
	Promedio réplicas	4,00E-06	2,18E-06	6,78E-06	2,46E-06
	Varianza réeplicas	1,04E-11	8,88E-42	1,05E-11	2,21E-14
	Desviación réplicas	3,22826E-06	2,97955E-21	3,2359E-06	1,48725E-07
	Tiempo computacional (Segundos)	629	850	1104	9506
10 Productos	Mejor Fitness encontrado en réplicas	3,67E-06	2,85E-06	2,88E-05	3,88E-06
	Promedio réplicas	1,18E-05	1,38E-05	0,00061584	4,19E-06
	Varianza réplicas	3,71E-10	1,49E-09	5,19E-07	1,40E-13
	Desviación réplicas	1,93E-05	3,86E-05	0,00072053	3,74E-07
	Tiempo computacional (Segundos)	1307	1613	1766	16943
20 Productos	Mejor Fitness encontrado en réplicas	1,55E-05	5,83E-06	0,00029021	N.A.
	Promedio réplicas	0,000242031	0,00012676	0,00495015	N.A.
	Varianza réplicas	1,10E-07	1,24E-07	1,69E-05	N.A.
	Desviación réplicas	0,000330947	0,00035167	0,00411664	N.A.
	Tiempo computacional (Segundos)	2402	2990	3548	N.A.
35 Productos	Mejor Fitness encontrado en réplicas	0,00059373	1,44E-05	0,00550722	N.A.
	Promedio réplicas	0,00609148	0,0014117	0,01916369	N.A.
	Varianza réplicas	1,75E-05	4,24E-06	0,00012615	N.A.
	Desviación replicas	0,00418052	0,0020603	0,01123181	N.A.
	Tiempo computacional (Segundos)	4194	5237	6056	N.A.
50 Productos	Mejor Fitness encontrado en réplicas	0,004966607	2,02E-05	0,0079255	N.A.
	Promedio réplicas	0,018249234	0,00283925	0,01313371	N.A.
	Varianza réplicas	9,93E-05	1,33E-05	5,77E-06	N.A.
	Desviación réplicas	0,009962958	0,003641995	0,00240261	N.A.
	Tiempo computacional (Segundos)	5472	7278	9047	N.A.

Tabla 6-5.: Tabla resumen de las medidas de desempeño de la función 4.

		F4			
		AG	PSO	CS	BFOA
5 Productos	Mejor Fitness encontrado en réplicas	23124,5	2233,6	2182,6	69529,7
	Promedio réplicas	125895,6	16385,1	6512,6	82084,4
	Varianza réplicas	2,66E+7	3,31E+7	3,13E+7	5,08E+7
	Desviación réplicas	51577,1	18220,8	5600,2	22551,57
	Tiempo computacional (Segundos)	208	421	711	2409
10 Productos	Mejor Fitness encontrado en réplicas	1187563,1	4989,2	303272	1540328,5
	Promedio réplicas	2424340,	325653,1	1023531,4	1657633,6
	Varianza réplicas	3,95E+11	1,61E+11	2,601E+11	7,31E+8
	Desviación réplicas	629277,39	401387,3	510030,4	27047,2
	Tiempo computacional (Segundos)	342	614	753	3462
20 Productos	Mejor Fitness encontrado en réplicas	4889430,7	59588,8	3255087,1	N.A.
	Promedio réplicas	7682892	2253100,4	5483552,8	N.A.
	Varianza réplicas	1,553E+12	2,59E+12	1,08E+12	N.A.
	Desviación réplicas	1247287,7	1610734	1042849,4	N.A.
	Tiempo computacional (Segundos)	617	1279	1541	N.A.
35 Productos	Mejor Fitness encontrado en réplicas	4345519,6	555379,9	4335549,3	N.A.
	Promedio réplicas	7122971,8	2652189,1	6024830,3	N.A.
	Varianza réplicas	1,308E+12	2,27E+12	7,35E+11	N.A.
	Desviación réplicas	1144051,06	1507672,02	859942,3	N.A.
	Tiempo computacional (Segundos)	1042	2105	2449	N.A.
50 Productos	Mejor Fitness encontrado en réplicas	12579656,53	1458810,568	9906973,59	N.A.
	Promedio réplicas	15762388,6	7049807,5	12771798,9	N.A.
	Varianza réplicas	2,76E+12	9,99E+12	1,066E+12	N.A.
	Desviación réplicas	1661763,8	3160698,6	1032461,3	N.A.
	Tiempo computacional (Segundos)	1345	3277	5026	N.A.

6.3. Aplicación de los algoritmos multi-objetivo

En la Tabla **6-6** se resumen las métricas, en el que los algoritmos, en cuanto al Tiempo Computacional el BCMOA para 5 y 10 productos es el mejor seguido del NSPSO, mientras que para 20, 35 y 50 productos el de mejor es el NSPSO, aunque el BCMOA tiene tiempos muy cercanos. El NSGA II y NSPSO son los de mayor diversidad indicando fronteras con mayor espaciamiento entre las soluciones, el BCMOA presenta en promedio una diversidad de 0,61 en todos los escenarios. En cuanto a la métrica de Dominancia de Pareto el NSGA II y BCMOA son los'unicos que aportan a la frontera final. Dado lo anterior se recomienda el NSGA -II y BCMOA para el problema propuesto.

En la Figura **6-9** se observan las fronteras generadas por los algoritmos en cada uno de los escenarios propuestos. Se observa que el NSGA II tiene una mayor cantidad de puntos óptimos, seguido por NSPSO y BCMOA. El MOCS no muestra un buen desempeño para este tipo de problema de inventario, dado los pocos puntos que convergen a la frontera.

Tabla 6-6.: Métricas encontradas los escenarios de 5, 10, 20, 35 y 50 productos

	5 Productos				**10 Productos**			
	NSGA II	NSPSO	BCMOA	MOCS	NSGA II	NSPSO	BCMOA	MOCS
Tiempo Computacional	93.13	42.14	34.47	64.49	97.0	56.77	52.68	124.56
Diversidad	1.29	0.82	0.58	0.48	1.51	0.78	0.55	0.71
Dominancia Pareto	0.50	0	0.49	0	0.4973	0	0.5027	0
	20 Productos				**35 Productos**			
Tiempo Computacional	113.51	82.14	90.17	244.36	136.09	111.8	128.26	420.82
Diversidad	1.48	0.99	0.68	0.90	1.39	1.02	0.62	0.54
Dominancia Pareto	0.4869	0	0.5131	0	0.5193	0	0.4807	0
	50 Productos							
Tiempo Computacional	160.78	148.59	199.45	595.24				
Diversidad	1.41	0.9222	0.6265	0.6160				
Dominancia Pareto	0.533	0	0.4667	0				

En la Figura **6-10** se muestra la frontera final y la alternativa seleccionada por TOPSIS como la solución ideal, teniendo en cuenta que a cada objetivo se le asignó el mismo peso de importancia.
Finalmente en la Tabla **6-7** se muestran los valores tomados en las funciones objetivo para la alternativa seleccionada. Al igual que en el capitulo anterior la función de costo de tranporte es la de mayor valor, por lo cual predomina a la hora de minimizar las otras funciones, esto se evidencia en la Figura **6-10** donde la Función 1 (costo de inventario y coordinación) se ubica en el máximo posible mientras que al mismo tiempo se localiza en el mínimo asumible para los objetivos 2,3 y 4.

Se observa que el modelo propuesto tiene un buen desempeño sin importar la cantidad a productos a coordinar, en el mismo sentido se recomienda el uso de NSGA II y BCMOA para solucionar este tipo de problemas de coordinación de inventario teniendo en cuenta demandas y tiempos de entrega estocásticos.

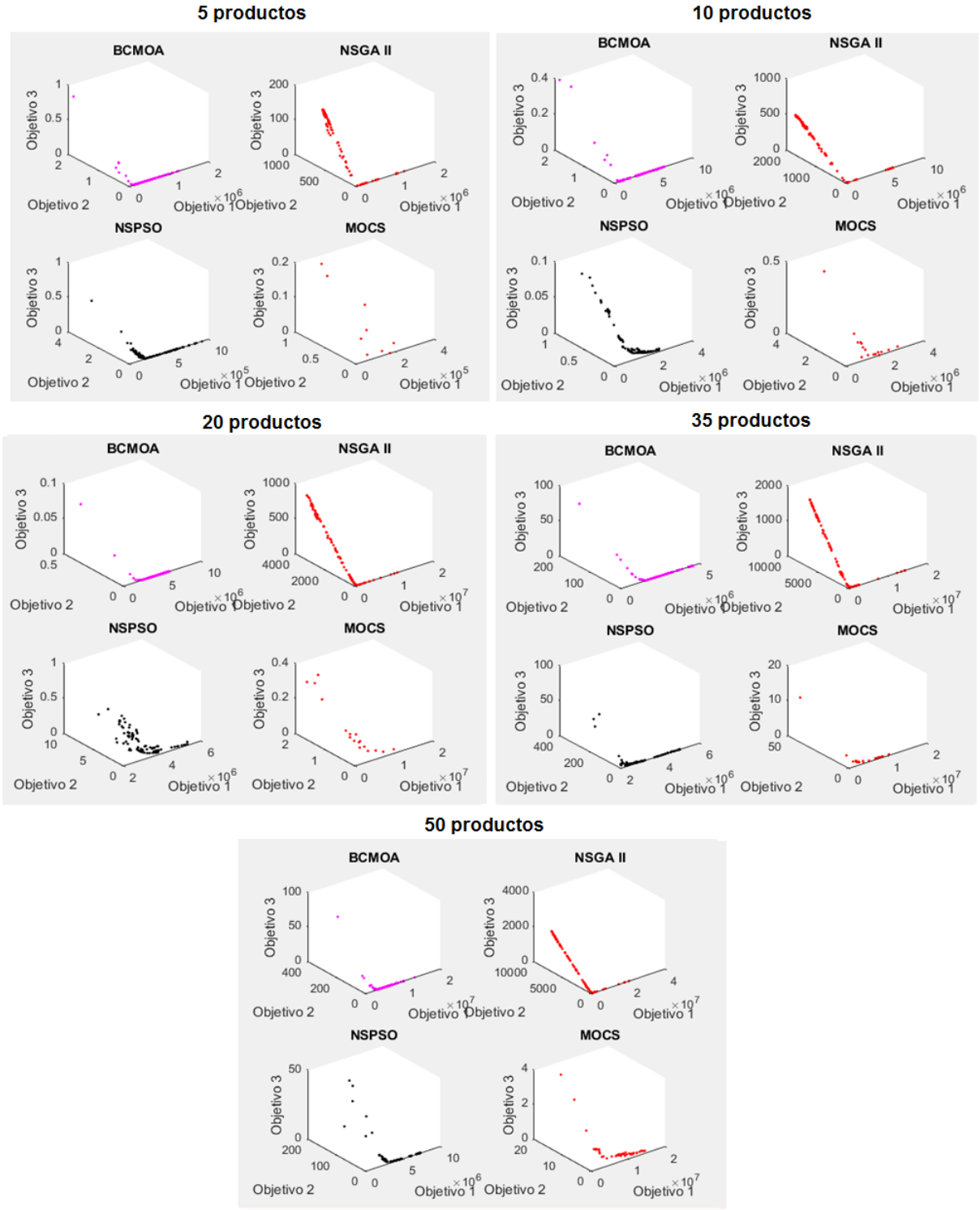

Figura 6-9.: Fronteras de paret o en cada escenario para cada algoritmo.

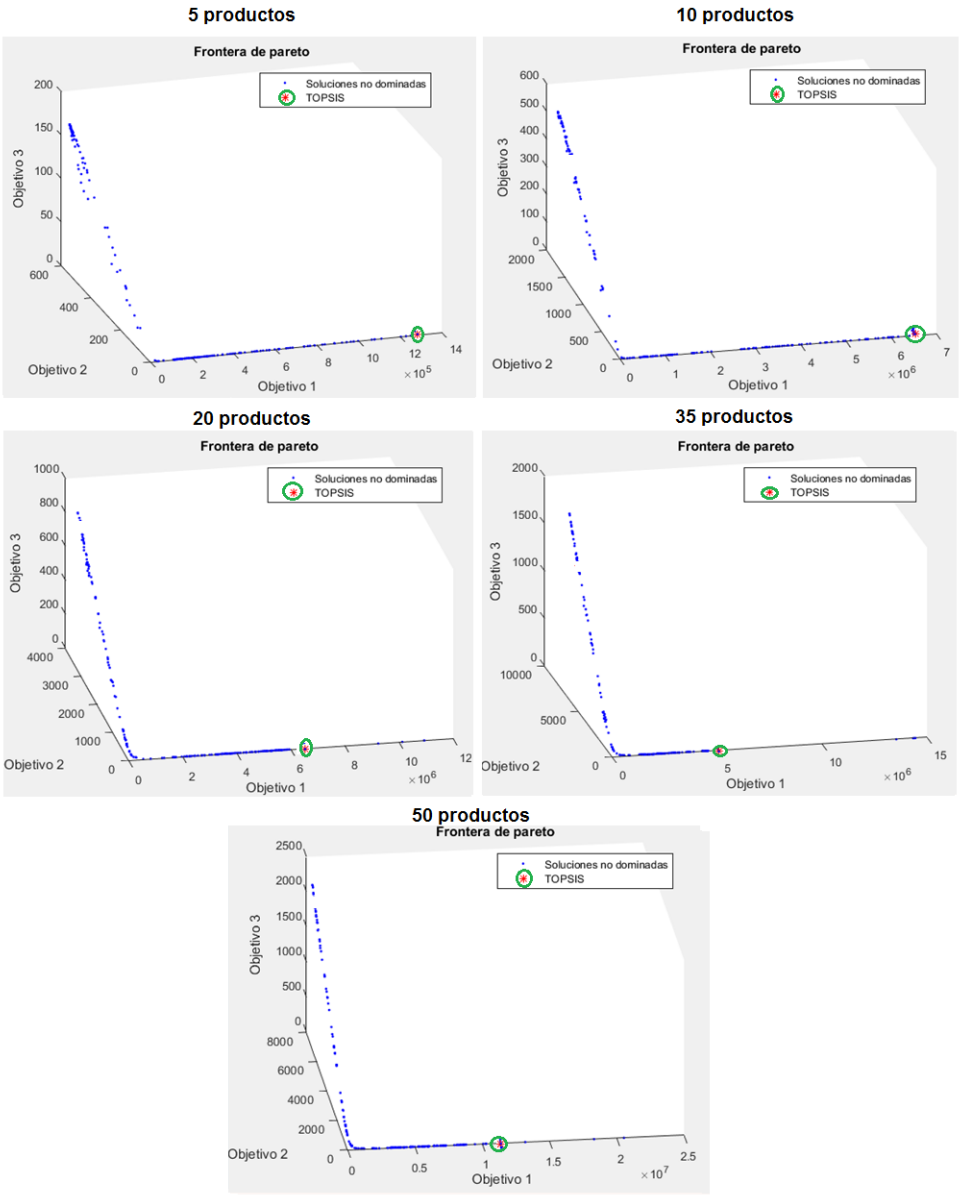

Figura 6-10.: Fronteras de Pareto final y alternativa selccionada mediante TOPSIS.

Tabla 6-7.: Valores de las funciones objetivo en la alternativa seleccionada en TOPSIS

	f1	f2	f3	f4
5 Productos	4331.4	395	125.6	5197.69
10 Productos	11932	1325.4	454.1	14318.4
20 Productos	34742.5	3044.7	871.2	29691.07
35 Productos	34965.8	5056.2	1523.5	541959.01
50 Productos	41862.6	6551.4	0.4667	1470235.2

7. Conclusiones y recomendaciones

7.1. Conclusiones

- La conclusión de mayor importancia es que las funciones de costos de inventario-coordinación y costos de transporte se movieron en la misma dirección, son altamente colineales, de forma análoga sucede con las funciones del nivel de servicio: Número de ocasiones en las que se tiene desabastecimiento y el número de unidades no satisfechas. Por lo anterior el problema resuelto con cuatro funciones objetivo se debe repensar con dos funciones objetivo.
- En la aplicación con datos reales se tiene que el uso de los algoritmos asume que el poder de transacción esta en el comprador. Lo que no es necesariamente cierto en el ecosistema, en especial en el sector salud.
- La conclusión anterior se debe generalizar, i.e. en el momento de aplicarse cualquier algoritmo se debe analizar el contexto de aplicación en pos de verificar la plausibilidad de los supuestos.
- En este trabajo se presentó un entorno multi-criterio para la toma de decisiones en la coordinación y control de inventarios con múltiples productos. En primer lugar se planteó un sistema multi-objetivo con cuatro funciones objetivo que buscan minimizar el costo pero a la vez maximizar los niveles de servicio con demandas y tiempos de entrega estocásticos.
- Aplicamos algoritmos bioinspirados tanto uni-objetivo y multi-objetivo existentes, los cuales se compararon con metricas de desempeño. Obteniendo que entre los algoritmos desarrollados el BCMOA y NSGA II fueron los de mejor desempeño dado su aporte a la conformación de la frotera de Pareto, en tiempos aceptables de solución y obteniendo una buena diversidad en las soluciones encontradas.
- Finalmente se aplicó TOPSIS para encontrar la solución ideal para cada uno de los escenarios propuestos y de esta manera facilitar la decisión de la politica de coordinación del inventario.

7.2. Recomendaciones

Este problema presenta una alta dimensionalidad, la cuál depende directamente del número de productos a hacer coordinados, siendo $(2 * n) + 1$ el número de variables del problema usando un portafolio que esta compuesto de n productos, por lo que una línea de investigación será disminuir esta dimensionalidad del problema a tratar. En trabajos futuros se pueden involucrar otras funciones objetivo tales manejo de riesgo en la cadena de suministro y economías de escala.

Este problema se puede extender a encontrar funciones que no dependan de la normalidad sino de otros tipos de distribución de probabilidad, permitiendo un acercamiento a casos reales.

7.2.1. Líneas de investigación

De lo presentado en el texto y lo analizado en un caso aplicado se debe introducir los siguientes elementos:

- Riesgo en la cadena de suministro.
- Asumir economías de escala en el momento de realizar un pedido.
- Introducir estacionalidad intra-anual en la demanda, es decir demanda que depende del mes observado, ejemplo de esto son los medicamentos utilizados en terapias respiratorias cuya rotación dependen de las temporadas de lluvia.
- Los algoritmos PSO y BFOA son desarrollados asumiendo que las variables a optimizar (Funciones objetivo) toman valores en todos los reales, análogo para la constante de coordinación pero en este trabajo se tiene que el número de ocasiones en las cuales se presentan desabastecimiento y el número de productos no satisfechos y la constante de coordinación sólo en los enteros. De donde un ítem para un trabajo de doctorado sería adaptar o postular algoritmos bio-inspirados para variables discretas.

Bibliografía

[1] Afsahi, Z. ; Javadzadeh ; R., Meybodi.: Hybrid Model of Particle Swarm Optimization and Cellular Learning Automata with New Structure of Neighborhood. En: *ICMLC 2011 3rd. International Conference on Machine Learning and Computing*, 2011

[2] Agrell, Per J.: A multicriteria framework for inventory control. En: *International Journal of Production Economics* 41 (1995), p. 59–70. – ISBN 0925–5273

[3] Arkin, Esther ; Joneja, Dev ; Roundy, Robin: Computational complexity of uncapacitated multi-echelon production planning problems. En: *Operations Research Letters* 8 (1989), Nr. 2, p. 61 – 66. – ISSN 0167–6377

[4] Ashlock, Daniel A.: *Evolutionary computation for modeling and optimization.* Springer, 2006. – I–XIX, 1–571 p.. – ISBN 978–0–387–22196–0

[5] Axsater, Sven: *Inventory Control.* Boston : Springer International Publishing, 2015 (International Series in Operations Research and Management Science). – ISBN 978–3–319–15729–0

[6] Castillo Tapia, M.G. ; Coello Coello, Carlos A.: Applications of multi-objective evolutionary algorithms in economics and finance: A survey. En: *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, p. 532–539

[7] Chopra, Sunil ; Meindl, Peter ; Pearson, Editorial (Ed.): *Administración de la cadena de suministro Estrategia, Planeación y Operación.* 3. 2008. – 552 p.. – ISBN 9789702611929

[8] Coello, C.A.C. ; Pulido, G.T. ; Lechuga, M.S.: Handling multiple objectives with particle swarm optimization. En: *Evolutionary Computation, IEEE Transactions on* 8 (2004), June, Nr. 3, p. 256–279. – ISSN 1089–778X

[9] Coello, Carlos A. C. ; Lamont, Gary B. ; Veldhuizen, David A. V.: *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation).* Secaucus, NJ, USA : Springer-Verlag New York, Inc., 2006. – ISBN 0387332545

[10] Coello Coello, Carlos A. ; Lechuga, M.S.: MOPSO: a proposal for multiple objective particle swarm optimization. En: *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on* Vol. 2, 2002, p. 1051–1056

[11] Dantzig, George B. ; Thapa, Mukund N.: *Linear Programming 1: Introduction.* Secaucus, NJ, USA : Springer-Verlag New York, Inc., 1997. – ISBN 0–387–94833–3

[12] Kap. Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications In: Das, Swagatam ; Biswas, Arijit ; Dasgupta, Sambarta ; Abraham, Ajith: *Foundations of Computational Intelligence Volume 3: Global Optimization*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2009, p. 23–55. – ISBN 978–3–642–01085–9

[13] Deb, K. ; Pratap, A. ; Agarwal, S. ; Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. En: *Evolutionary Computation, IEEE Transactions on* 6 (2002), Apr, Nr. 2, p. 182–197. – ISSN 1089–778X

[14] Deb, Kalyanmoy ; Kalyanmoy, Deb: *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA : John Wiley & Sons, Inc., 2001. – ISBN 047187339X

[15] Ehrgott, Matthias (Ed.) ; Gandibleux, Xavier (Ed.): *Multiple criteria optimization : state of the art annotated bibliographic surveys*. Boston : Kluwer Academic Publishers, 2002 (International series in operations research & management science). – ISBN 1–402–07128–0

[16] Eichfelder, Gabriele: *Adaptive scalarization methods in multiobjective optimization*. 2008. – 241 S. p.. – ISBN 978–3–540–79157–7\r978–3–540–79159–1 (eBook)

[17] Eijs, M. J. G. v.: A Note on the Joint Replenishment Problem under Constant Demand. En: *The Journal of the Operational Research Society* 44 (1993), Nr. 2, p. pp. 185–191. – ISSN 01605682

[18] Gallardo Castro, Victor A.: *Análisis del consumo de medicamentos incluidos en los tratamientos de patologías pertenecientes al régimen general de garantías explícitas en salud en el hospital de corral durante el año 2005*, Universidad austral de chile, Tesis de Grado, 2008. – 95 p.

[19] Goh, Chi-Keong ; Tan, Kay C.: A Competitive-cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization. En: *Trans. Evol. Comp* 13 (2009), Februar, Nr. 1, p. 103–127. – ISSN 1089–778X

[20] Goldberg, David E.: *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1989. – ISBN 0201157675

[21] Goyal, S. K.: Determination of Optimum Packaging Frequency of Items Jointly Replenished. En: *Management Science* 21 (1974), Nr. 4, p. 436–443. – ISSN 00251909, 15265501

[22] Guzmán, M.: *Técnicas de otimização baseadas em quimiotaxia de bactérias*, Escola de Engenharia de São Carlos, Universidade de São Paulo, Tesis de Grado, 2009

[23] Guzmán, María A. ; Delgado, Alberto ; Carvalho, Jonas D.: A novel multiobjective optimization algorithm based on bacterial chemotaxis. En: *Engineering Applications of Artificial Intelligence* 23 (2010), Nr. 3, p. 292 – 301. – ISSN 0952–1976

[24] Harris, Ford W.: How Many Parts to Make at Once. En: *The Magazine of Management* 10 (1913), Nr. 2, p. 135–136

[25] Holland, John H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. University of Michigan Press, 1975. – ISBN 0262082136

[26] Hwang, C. L., & Yoon, K.: *Multiple Objective Decision Making Methods and Applications*. 1979. – 358 p.. – ISBN 9781439861578

[27] James H. Bookbinder, Vincent Y. X. C.: Multicriteria Trade-Offs in a Warehouse/Retailer System. En: *The Journal of the Operational Research Society* 43 (1992), Nr. 7, p. 707–720. – ISSN 01605682, 14769360

[28] Kennedy, J ; Eberhart, R: Particle swarm optimization. En: *Neural Networks, 1995. Proceedings., IEEE International Conference on* 4 (1995), p. 1942–1948 vol.4. – ISBN VO – 4

[29] Khouja, Moutaz ; Goyal, Suresh: A review of the joint replenishment problem literature: 1989-2005. En: *European Journal of Operational Research* 186 (2008), Nr. 1, p. 1–16. – ISSN 03772217

[30] Khouja, Moutaz ; Goyal, Suresh: A review of the joint replenishment problem literature: 1989-2005. En: *European Journal of Operational Research* 186 (2008), Nr. 1, p. 1 – 16. – ISSN 0377–2217

[31] Khouja, Moutaz ; Michalewicz, Zbigniew ; Satoskar, Sandeep S.: A comparison between genetic algorithms and the RAND method for solving the joint replenishment problem. En: *Production Planning & Control* 11 (2000), Nr. March 2015, p. 556–564. – ISSN 0953–7287

[32] Lai, Young-Jou ; Liu, Ting-Yun ; Hwang, Ching-Lai: TOPSIS for MODM. En: *European Journal of Operational Research* 76 (1994), Nr. 3, p. 486 – 500. – Facility Location Models for Distribution Planning. – ISSN 0377–2217

[33] Lamban, Ma PILAR et a.: Modelo para el cálculo del costo de almacenamiento de un producto: Caso de estudio en un entorno logístico. En: *DYNA* 80 (2013), 06, p. 23 – 32. – ISSN 0012–7353

[34] Li, Xiaodong: A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. En: *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: PartI*. Berlin, Heidelberg : Springer-Verlag, 2003 (GECCO'03). – ISBN 3–540–40602–6, p. 37–48

[35] Mahapatra, N.K. ; Maiti, M.: Decision process for multiobjective, multi-item production-inventory system via interactive fuzzy satisficing technique. En: *Computers and Mathematics with Applications* 49 (2005), Nr. 5-6, p. 805 – 821. – ISSN 0898–1221

[36] Mandal, Nirmal K. ; Roy, Tapan K. ; Maiti, Manoranjan: Multi-objective fuzzy inventory model with three constraints: a geometric programming approach. En: *Fuzzy Sets and Systems* 150 (2005), Nr. 1, p. 87 – 106. – ISSN 0165–0114

[37] Mora, Hector: *Programación Lineal*. 2. 2004. – 282 p.

[38] Moslemi, H. ; Zandieh, M.: Comparisons of some improving strategies on {MOPSO} for multi-objective (r, Q) inventory system. En: *Expert Systems with Applications* 38 (2011), Nr. 10, p. 12051 – 12057. – ISSN 0957–4174

[39] Mousavi, Seyed M. ; Sadeghi, Javad ; Niaki, Seyed Taghi A. ; Tavana, Madjid: A bi-objective inventory optimization model under inflation and discount using tuned Pareto-based algorithms: NSGA-II, NRGA, and {MOPSO}. En: *Applied Soft Computing* 43 (2016), p. 57 – 72. – ISSN 1568–4946

[40] Padmanabhan, G. ; Vrat, Prem: Analysis of multi-item inventory systems under resource constraints: A non-linear goal programming approach. En: *Engineering Costs and Production Economics* 20 (1990), Nr. 2, p. 121 – 127. – ISSN 0167–188X

[41] Park, KyoungJong ; Kyung, Gyouhyung: Optimization of total inventory cost and order fill rate in a supply chain using PSO. En: *The International Journal of Advanced Manufacturing Technology* 70 (2013), Nr. 9, p. 1533–1541. – ISSN 1433–3015

[42] Pasandideh, Seyed Hamid R. ; Niaki, Seyed Taghi A. ; Sharafzadeh, Sharareh: Optimizing a bi-objective multi-product EPQ model with defective items, rework and limited orders: NSGA-II and MOPSO algorithms. En: *Journal of Manufacturing Systems* 32 (2013), Nr. 4, p. 764 – 770. – ISSN 0278–6125

[43] Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. En: *Control Systems, IEEE* 22 (2002), Jun, Nr. 3, p. 52–67. – ISSN 1066–033X

[44] Poole, David ; Mackworth, Alan ; Goebel, Randy: *Computational Intelligence: A Logical Approach*. Oxford, UK : Oxford University Press, 1997. – ISBN 0–19–510270–3

[45] Prawda, J ; Limusa (Ed.): *Métodos y modelos de investigación de operaciones*. 3ra Edició. Limusa, 1994. – 200 p.. – ISBN 9681805909

[46] Puerto, J. ; FernÃ¡ndez, F.R.: Pareto-optimality in classical inventory problems. En: *Naval Research Logistics (NRL)* 45 (1998), Nr. 1, p. 83–98. – ISSN 1520–6750

[47] Robinson, J. ; Rahmat-Samii, Y.: Particle swarm optimization in electromagnetics. En: *IEEE Transactions on Antennas and Propagation* 52 (2004), Feb, Nr. 2, p. 397–407. – ISSN 0018–926X

[48] Roy, T.K. ; Maiti, M.: Multi-objective inventory models of deteriorating items with some constraints in a fuzzy environment. En: *Computers and Operations Research* 25 (1998), Nr. 12, p. 1085 – 1095. – ISSN 0305–0548

[49] Russell, Stuart J. ; Norvig, Peter: *Artificial Intelligence: A Modern Approach*. 2. Pearson Education, 2003. – ISBN 0137903952

[50] Sarker, Ruhul A. ; Newton, Charles S. ; Press, CRC (Ed.): *Optimization Modelling: A Practical Approach*. 2007. – 502 p.. – ISBN 9781420043105

[51] Shi, Y ; Eberhart, R.: A modified particle swarm optimizer. En: *IEEE International Conference on Evolutionary Computation*, 1998

[52] Silver, Edward ; Pyke, David ; Peterson, Rein ; John Wiley & Sons (Ed.): *Inventory Management and Production Planning and Scheduling*. 3. John Wiley & Sons, 1998. – 784 p.. – ISBN 0471119474

[53] Srinivas, N. ; Deb, Kalyanmoy: Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. En: *Evol. Comput.* 2 (1994), September, Nr. 3, p. 221–248. – ISSN 1063–6560

[54] Kap. Multi Objective Cuckoo Search Optimization for Fast Moving Inventory Items In: Srivastav, Achin ; Agrawal, Sunil: *Advances in Intelligent Informatics*. Cham : Springer International Publishing, 2015, p. 503–510. – ISBN 978–3–319–11218–3

[55] Triantaphyllou, E.: *Applied Optimization*. Vol. 44: *Multi-Criteria Decision Making Methodologies: A Comparative Study*. Dordrecht, 2000. – 320 pages p.

[56] Tsou, Ching S.: Multi-objective inventory planning using MOPSO and TOPSIS. En: *Expert Systems with Applications* 35 (2008), Nr. 1-2, p. 136–142. – ISBN 0957–4174

[57] Tsou, Ching-Shih: Evolutionary Pareto optimizers for continuous review stochastic inventory systems. En: *European Journal of Operational Research* 195 (2009), Nr. 2, p. 364 – 371. – ISSN 0377–2217

[58] Tzeng, Gwo-Hshiung. ; Huang, Jih-Jeng: *Multiple Attribute Decision Making: Methods and Applications*. 1st. Chapman and Hall,CRC, 2011

[59] Winston, Waine ; Thomson Internacional (Ed.): *Investigación de operaciones*. 4. 2004. – 1440 p.. – ISBN 970–695–362–1

[60] Yang, X. S. ; Deb, S.: Cuckoo search via Lévy flights. En: *World Congr. Nat. Biol. Inspired Comput. NABIC 2009*, 2009, p. 210–214

[61] Yang, Xin-She ; Deb, Suash: Engineering Optimisation by Cuckoo Search. (2010), p. 1–17

[62] Yang, Xin-She ; Deb, Suash: Multiobjective Cuckoo Search for Design Optimization. En: *Comput. Oper. Res.* 40 (2013), Juni, Nr. 6, p. 1616–1624. – ISSN 0305–0548

[63] Yousefi, Ommolbanin ; Aryanezhad, Mirbahadorgholi ; Sadjadi, Seyed J. ; Shahin, Arash: Developing a multi-objective, multi-item inventory model and three algorithms for its solution. En: *Journal of Zhejiang University SCIENCE C* 13 (2012), Nr. 8, p. 601–612. – ISSN 1869–1951

A. Anexo: Deducción del modelo propuesto

A.0.1. Función de costo de inventario y pedido coordinado

Dado que las variables del modelo JRP son T y k_i y las de los modelos de inventarios estocáticos son normalmente Q_i y fs, se busca dejar la ecuaciones en términos de Q y SS. Sabiendo que $T_i = k_i T$ y $Q_i = D_i T_i$

$$\Psi(\varrho(T, k_i)) = \frac{T}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{1}{T}(S + \sum_{i=1}^{n}\frac{s_i}{k_i}) \tag{A-1}$$

$$= \frac{\frac{T_i}{k_i}}{2}\sum_{i=1}^{n} k_i D_i h_i + \frac{1}{\frac{T_i}{k_i}}(S + \sum_{i=1}^{n}\frac{s_i}{k_i}) \tag{A-2}$$

$$= \sum_{i=1}^{n}\frac{T_i}{2 * k_i} k_i D_i h_i + (\sum_{i=1}^{n}\frac{k_i}{T_i} S + \sum_{i=1}^{n}\frac{k_i}{T_i}\frac{s_i}{k_i}) \tag{A-3}$$

$$= \sum_{i=1}^{n}\frac{\frac{Q_i}{D_i}}{2 * k_i} k_i D_i h_i + (\sum_{i=1}^{n}\frac{k_i}{\frac{Q_i}{D_i}} S + \sum_{i=1}^{n}\frac{k_i}{\frac{Q_i}{D_i}}\frac{s_i}{k_i}) \tag{A-4}$$

$$= \sum_{i=1}^{n}\frac{Q_i}{2} h_i + \sum_{i=1}^{n} S\frac{k_i D_i}{Q_i} + \frac{s_i D_i}{Q_i} \tag{A-5}$$

$$= \sum_{i=1}^{n}\frac{Q_i}{2} h_i + \sum_{i=1}^{n}\frac{D_i}{Q_i}(S k_i + s_i) \tag{A-6}$$

$$\Psi(\phi(Q_i, k_i)) = \sum_{i=1}^{n}\frac{Q_i}{2} h_i + \sum_{i=1}^{n}\frac{D_i}{Q_i}(S k_i + s_i) \tag{A-7}$$

Como en la ecuación 2-21 no se tiene en cuenta la demanda aleatoria durante el tiempo de entrega, se procede a agregar un factor de seguridad (ks_i), este factor afecta el costo de inventario:

$$\sum_{i=1}^{n}\left(\frac{Q_i}{2} + ks_i * \sigma_{DLi}\right) * h_i \tag{A-8}$$

Por lo que la función de costo será:

$$(f_1)\ \Psi(\delta(Q_i, k_i, ks_i)) = \left(\sum_{i=1}^{n}\left(\frac{Q_i}{2} + ks_i * \sigma_{DLi}\right) * h_i\right) + \left(\sum_{i=1}^{n}\frac{E(D_i)}{Q_i}(Sk_i + s_i)\right) \tag{A-9}$$

A.0.2. Función de frecuencia esperada de ocasiones de desabastecimiento anuales.

Se tiene la siguiente función propuesta en Tsou:

$$Min\, N(Q, ks) = \frac{D}{Q}\int_{r}^{\infty} f_{DL}(x)dx \tag{A-10}$$

$$= \frac{D}{Q}(1 - \Phi(ks)) \tag{A-11}$$

donde Φ es la función de probabilidad acumulada de la distribución normal. Si la función anterior la adaptamos para multiobjetivo y teniendo en cuenta la notación tendremos:

$$(f_2)\ \Psi(\vartheta(Q_i, ks_i) = \sum_{i=1}^{n}\frac{E(D_i)}{Q_i}(1 - \Phi(ks_i)) \tag{A-12}$$

A.0.3. Función de la cantidad esperada de productos faltantes anualmente

Se tiene que:

$$Min\, S(Q, k) = \frac{D}{Q}\int_{r}^{\infty}(x - r)\, f_{DL}(x)dx = \frac{D\sigma_{DL}}{Q}(\phi(ks) - ks(1 - \Phi(ks))) \tag{A-13}$$

donde Φ es la función de probabilidad acumulada de la distribución normal, ϕ es la función de probabilidad de la distribución normal . Si la función anterior la adaptamos para multiobjetivo tendriamos:

$$(f_3)\ \Psi(\varphi(Q_i, ks_i) = \sum_{i=1}^{n}\left[\frac{E(D_i)\sigma_{DL_i}}{Q_i}(\phi(ks_i) - ks_i(1 - \Phi(ks_i)))\right] \tag{A-14}$$

A.0.4. Función de costo de transporte del inventario

La función propuesta por Youseefi, se deja en términos de la variable Q_i.

$$\Psi(\Delta(T, k_i)) = T \sum_{i=1}^{n} tr_i k_i D_i \tag{A-15}$$

$$= \frac{T_i}{k_i} \sum_{i=1}^{n} tr_i k_i D_i \tag{A-16}$$

$$= \frac{\frac{Q_i}{D_i}}{k_i} \sum_{i=1}^{n} tr_i k_i D_i \tag{A-17}$$

$$= \sum_{i=1}^{n} tr_i Q_i \tag{A-18}$$