Centro de Enseñanza LANIA

Comparativo Empírico de Algoritmos Evolutivos Multiobjetivo en la Reducción del Efecto Látigo y Costos Totales en Cadenas de Suministro

TESIS

Que presenta:
Elyar Alberto López Dávila

Para obtener el grado de:
Maestro en Computación Aplicada

Directores de Tesis:
Dr. Efrén Mezura Montes
Dr. Luis Ernesto Mancilla Espinosa
Dra. Cora Beatriz Excelente Toledo

Xalapa, Veracruz, México Febrero 2015

Agradecimientos

A mi familia. A mis padres Elsa y Arnulfo, a mis hermanas Aolani y Mariam, gracias por el apoyo, los consejos y el amor que siempre me han brindado. A mis abuelos Lucio, Ana, Arnulfo y Mercedes, una parte de ustedes siempre está conmigo. A mi tía Ofelia, muchas gracias por su hospitalidad de todos estos años.

A todos mis compañeros de maestría, gracias por apoyarme y acompañarme durante dos años de trabajo y por hacer esos momentos difíciles más llevaderos, pero sobre todo por su amistad.

Al Dr. Efrén Mezura Montes por su apoyo, consejos y excelente guía para la culminación de este trabajo de tesis.

Al Dra. Cora Beatriz Excelente Toledo por sus consejos, comentarios y la motivación brindada para concluir este trabajo de tesis.

Al Dr. Luis Ernesto Mancilla Espinosa por permitirme realizar una estancia académica en el Instituto Tecnológico de León.

A todos mis profesores de la maestría, Mtro. Pedro, Mtro. Julio, Dr. Horacio, Mtra. Betty, Dra. Margarita, Dr. Eduardo, Mtra. Yesenia, Mtro. Juan, Mtro. Ocharán y Dr. Alejandro.

A todo el personal administrativo de LANIA, cuya encomiable labor siempre fue de gran ayuda.

Al CONACyT por la beca otorgada para la realización de la maestría.

Resumen

En este trabajo de tesis se presenta la resolución de una instancia del problema de la cadena de suministro, el cual está modelado como un problema multiobjetivo con restricciones, donde se requiere minimizar los costos totales de operación y el efecto látigo.

Como propuesta de solución se utilizó el algoritmo de optimización basado en el forrajeo de bacterias multiobjetivo (MOMBFOA) combinado con el operador de mutación diferencial, propio de Evolución Diferencial (ED). Los resultados de este algoritmo son comparados con otros arrojados por algoritmos del estado del arte (NSGA-II y SPEA2) mediante las métricas de desempeño Two Set Coverage e Hypervolume y se validó si existían diferencias significativas entre ellos mediante la prueba de Wilcoxon.

La propuesta de solución utilizada logra resolver de manera satisfactoria la instancia del problema de la cadena de suministros y arroja resultados competitivos, e incluso mejores en algunas métricas, con respecto a los otros dos algoritmos de optimización multiobjetivo del estado del arte.

Índice general

1. Introducción **1**
- 1.1. Planteamiento del problema . . . 2
- 1.2. Objetivos . . . 2
  - 1.2.1. Objetivo general . . . 2
  - 1.2.2. Objetivos específicos . . . 3
- 1.3. Hipótesis . . . 3
- 1.4. Justificación . . . 3
- 1.5. Contribuciones . . . 4

2. Optimización **5**
- 2.1. Elementos del problema del diseño óptimo . . . 5
  - 2.1.1. Variables de diseño . . . 7
  - 2.1.2. Restricciones . . . 7
  - 2.1.3. Función objetivo . . . 7
  - 2.1.4. Límites de las variables . . . 8
- 2.2. Optimización multiobjetivo . . . 8
  - 2.2.1. Definición formal . . . 9
  - 2.2.2. Óptimo de Pareto . . . 9
  - 2.2.3. Dominancia de Pareto . . . 10
  - 2.2.4. Frente de Pareto . . . 11

3. Cadena de suministro **13**
- 3.1. Efecto látigo . . . 14
- 3.2. Optimización en cadenas de suministro . . . 16
- 3.3. Modelo formal de minimización de costos totales y efecto látigo . 18

4. MOMBFOA **21**
- 4.1. Algoritmos de optimización basados en inteligencia colectiva . . . 21
- 4.2. BFOA . . . 22
- 4.3. MBFOA . . . 24
- 4.4. MOMBFOA . . . 25
  - 4.4.1. Dominancia de Pareto y mecanismo de manejo de restricciones 27
  - 4.4.2. Archivo externo . . . 27

4.4.3. Mecanismo de diversidad 27
4.4.4. Operador de atracción 27
4.4.5. Reproducción 29
4.5. Híbrido MOMBFOA - DE 31

5. Experimentación **34**
5.1. NSGA-II 34
5.1.1. Ordenamiento rápido no dominado 34
5.1.2. Preservación de la diversidad 35
5.1.3. Ciclo principal 35
5.2. SPEA2 38
5.2.1. Asignación de fitness 39
5.2.2. Selección ambiental 40
5.3. Métricas de desempeño 41
5.3.1. Two-Set Coverage 41
5.3.2. Hypervolume 41
5.4. Diseño experimental 41
5.5. Comparación de resultados 42
5.6. Discusión de resultados 44

6. Conclusiones y trabajos futuros **48**

Bibliografía **54**

Índice de figuras

1.1. Modelo de la instancia del problema de la cadena de suministro . 3

2.1. Diagrama de flujo del proceso de diseño óptimo 6
2.2. Principio de dualidad 8
2.3. Óptimo de Pareto 10
2.4. Dominancia de Pareto 11
2.5. Frente de Pareto 12

3.1. Ejemplo de cadena de suministro 14
3.2. Efecto látigo 16

4.1. Pseudocódigo de BFOA 23
4.2. Pseudocódigo de MBFOA 26
4.3. Asignación de distancia crowding 28
4.4. Cálculo de la distancia crowding 28
4.5. Pseudocódigo de MOMBFOA 30
4.6. Cruza de Evolución Diferencial 32
4.7. Pseudocódigo de MOMBFOA-DE 33

5.1. Ordenamiento rápido no dominado 36
5.2. Procedimiento del NSGA-II 37
5.3. Ciclo principal de NSGA-II 37
5.4. Pseudocódigo de SPEA2 39
5.5. Método de truncado en SPEA2 40
5.6. Hypervolume 42
5.7. Frentes acumulados de 30 corridas 45
5.8. Sección de los frentes acumulados 47

Índice de tablas

3.1. Variables y parámetros del modelo modificado 19
3.2. Valores de los parámetros 20

5.1. Valores de los parámetros de los algoritmos utilizados 43
5.2. Estadísticas sobre los valores de Hypervolume obtenidos de las 30 corridas de cada algoritmo 44
5.3. Prueba de Wilcoxon a valores de Hypervolume. 44
5.4. Estadísticas sobre los valores de Two-Set Coverage obtenidos de las 30 corridas de cada algoritmo. 44
5.5. Valores de Hypervolume de los frentes acumulados 45
5.6. Valores de Two-Set Coverage de los frentes acumulados 45
5.7. Valores de las funciones objetivo de las tres soluciones extraídas de los frentes acumulados. 46

Capítulo 1

Introducción

La optimización se define como el proceso mediante el cual se busca encontrar los valores de decisión que minimicen o maximicen el valor de una función, dentro de un espacio de búsqueda determinado [10].

Hoy en día, dentro de las diversas actividades realizadas por el ser humano, se encuentran problemas de optimización, algunos muy complejos que no pueden ser resueltos con técnicas matemáticas clásicas [46].

Uno de estos problemas complejos es aquel relacionado con la cadena de suministro, la cual se define como un conjunto de funciones, procesos y actividades que permiten que la materia prima, productos o servicios sean transformados, entregados y consumidos por el cliente final [30].

Estos problemas son difíciles de resolver debido a la variación de los diversos factores, interacción entre las entidades, la longitud de la cadena de suministro, los tiempos de fabricación y envío, la complejidad del modelo, etcétera. Todo ello modifica el comportamiento de la cadena de suministro y una mala decisión por parte de los primeros niveles (los que tratan directamente con el cliente final) repercute de manera significativa sobre el fabricante.

Las técnicas clásicas de optimización, como programación lineal, programación entera, programación con restricciones provenientes de la investigación de operaciones intentan resolver el escenario en forma parcial con resultados satisfactorios en ciertas instancias donde las características del problema las hacen aplicables [12].

Es por ello que las técnicas inspiradas en la naturaleza (algoritmos evolutivos, inteligencia colectiva) se vuelven una herramienta fundamental para tratar de resolver instancias de problemas de la cadena de suministro donde los métodos tradicionales no son una alternativa viable.

En este dominio, existen dos factores importantes para medir el desempeño del diseño de una cadena de suministro: el costo total de operaciones y el llamado efecto látigo. Este último es un fenómeno que se da cuando los diversos actores dentro de una cadena de suministro realizan pedidos, los cuales por beneficio

particular o especulación de la demanda, son distorsionados; ello hace que la demanda real en el punto de venta, produzca un gran movimiento en el extremo de la cadena, donde está el fabricante, lo que provoca problemas como: incremento innecesario del inventario, incremento en los costos de almacenamiento y retraso en la entrega del pedido, lo que en consecuencia deriva en una deficiente gestión de la cadena [24].

Recientemente se ha propuesto un modelo de cadena de suministro en el cual los objetivos a optimizar son dos: la reducción efecto látigo y el costo total de operaciones [2]. Si bien este trabajo citado exploró este tema, lo hizo mediante dos algoritmos únicamente, es claro que un mayor estudio se requiere para explorar diversas técnicas que potencialmente pueden resolver este tipo de problemas y dar resultados competitivos o mejores, lo cual se pretende con el presente trabajo.

1.1. Planteamiento del problema

El problema que se resolverá en esta tesis es un problema de optimización multiobjetivo. De ahí que se presente su definición.

La optimización multiobjetivo es aquella en la cual se optimizan dos o más funciones objetivo, las cuales describen un problema. Esto se trata de buscar los valores de las variables o parámetros de un problema que hagan que los valores de las funciones objetivo sean óptimos y que satisfagan un conjunto de restricciones de desigualdad y de igualdad [25].

Adicionalmente, las funciones objetivo a optimizar se encuentran en conflicto, esto quiere decir que si algunos valores de las variables del problema mejoran una función objetivo, empeorarán las otras.

En este trabajo de tesis se ataca una instancia del problema de la cadena de suministro, donde lo que se busca es minimizar los costos totales al mismo tiempo que se disminuye el efecto látigo. La instancia consta de dos escalones: el distribuidor y el mayorista, los cuales intercambian distintos producto entre sí. Una explicación gráfica del problema puede observarse en la Fig. 1.1.

1.2. Objetivos

1.2.1. Objetivo general

Comparar el desempeño de tres algoritmos de optimización multiobjetivo: NSGA-II [18], SPEA2 [19], MOMBFOA [36] (combinado con el operador de mutación diferencial), para resolver una instancia de problema multiobjetivo de una cadena de suministro, donde se busca minimizar el efecto látigo y el costo total de operaciones.

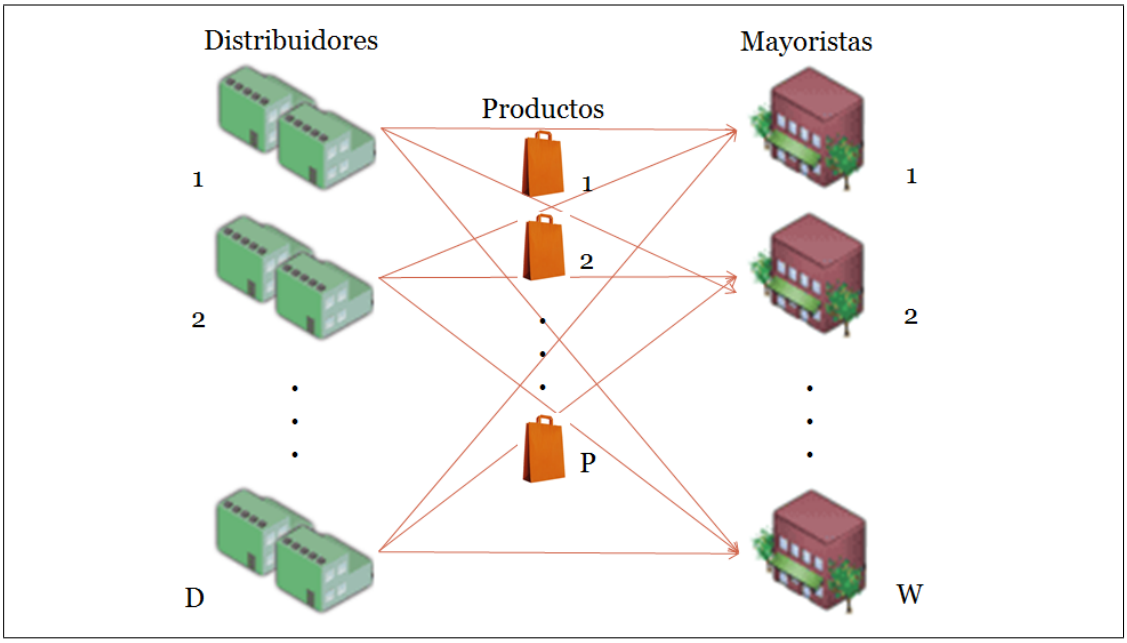

Figura 1.1: Modelo de la instancia del problema de la cadena de suministro abordado en esta tesis.

1.2.2. Objetivos específicos

- Revisar algoritmos de optimización utilizados para resolver problemas de optimización multiobjetivo.
- Implementar tres algoritmos de optimización multiobjetivo para la resolución de una instancia multiobjetivo del problema de la cadena de suministro.
- Medir el desempeño de los algoritmos utilizados mediante métricas propias de los problemas multiobjetivo: Two-Set coverage e Hypervolume [36].

1.3. Hipótesis

El algoritmo MOMBFOA combinado con el operador de mutación diferencial obtendrá un desempeño mejor que los otros algoritmos comparados, en las soluciones finales y/o en los valores de las métricas de desempeño (Two-Set coverage e Hypervolume), en una instancia del problema de la cadena de suministro donde los objetivos a minimizar son el efecto látigo y los costos totales de operación.

1.4. Justificación

Los problemas relacionados con la cadena de suministro, por lo regular son de una complejidad significativa, debido a la interacción entre las entidades, la longi-

tud de la cadena de suministro, los tiempos de fabricación y envío, la complejidad del modelo, entre otros. Es por ello que se hace necesario contar con técnicas que permitan resolver las diversas instancias que se presentan en la gestión de las cadenas de suministro.

El problema que se aborda es una propuesta reciente [2], ya que únicamente se han reportado soluciones de dos algoritmos y no se han explorado otras técnicas que potencialmente podrían dar resultados competitivos o mejores.

Por otro lado, debido a que el problema a resolver se modela como un problema multiobjetivo, se puede atacar mediante técnicas meta-heurísticas, como un algoritmo bio-inspirado, las cuales poseen la ventaja de poder manejar simultáneamente un conjunto de resultados en una sola ejecución del algoritmo dando diversas soluciones al problema. Además estas técnicas tienen menos problemas al aplicarse en espacios de búsqueda complejos en comparación con técnicas matemáticas clásicas.

1.5. Contribuciones

1. El primer estudio comparativo de tres algoritmos bio-inspirados multiobjetivo en la resolución de una instancia particular del problema de la cadena de suministro donde se considera la disminución de los costos totales y el efecto látigo.

2. La primera versión adaptada del algoritmo MOMBFOA para resolver problemas de cadenas de suministro, la cual consiste en una combinación de los mecanismos de búsqueda de MOMBFOA y Evolución Diferencial.

Capítulo 2

Optimización

Optimización se puede definir como el proceso por el cual se busca minimizar o maximizar el valor de una función, eligiendo sistemáticamente los valores de las variables de decisión, dentro de un espacio de búsqueda permisible[51].

La optimización busca encontrar los máximos o mínimos posibles de una función, para lo cual se utilizan métodos que se conocen como técnicas de programación matemática, las cuales se aplican para encontrar las mejores soluciones posibles.

2.1. Elementos del problema del diseño óptimo

En diversas actividades de diseño dentro de la industria e ingeniería se lleva a cabo una búsqueda de un diseño óptimo, el proceso que usualmente se lleva a cabo en esta búsqueda es el mostrado en la Fig. 2.1. Para ello, es posible la utilización de diversas técnicas de optimización las cuales según sus características se pueden clasificar en[25]:

1. Basándose en la existencia de restricciones
    - Con restricciones
    - Sin restricciones
2. Basándose en la naturaleza de las variables de decisión
    - Problemas estáticos o paramétricos
    - Problemas dinámicos o de trayectorias (las variables son función de un parámetro determinado)
3. Basándose en la naturaleza de las ecuaciones involucradas
    - Problemas lineales

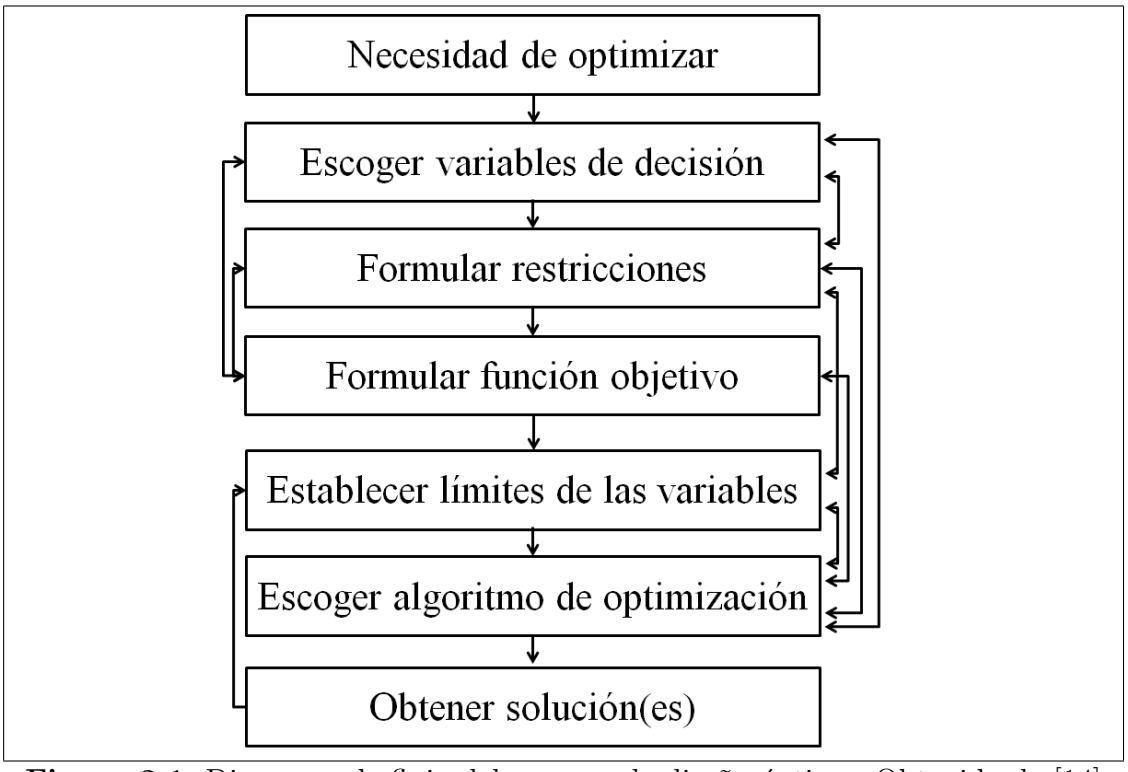

Figura 2.1: Diagrama de flujo del proceso de diseño óptimo. Obtenido de [14]

- Problemas no lineales
- Problemas de programación geométrica
- Problemas de programación cuadrática

4. Basándose en los valores permisibles de las variables de diseño
   - Problemas de programación entera
   - Problemas de programación con valores reales
   - Problemas donde se busca un orden óptimo de elementos (optimización combinatoria)
5. Basándose en la naturaleza determinista de las variables
   - Problemas estocásticos
   - Problemas deterministas
6. Basándose en la separabilidad de las funciones
   - Problemas separables
   - Problemas no separables

7. Basándose en el número de funciones objetivo
    - Mono-objetivo
    - Multiobjetivo

Algunos aspectos del proceso del diseño óptimo se definen a continuación[14].

2.1.1. Variables de diseño

Un problema de diseño involucra, por lo regular, varios parámetros de diseño, de los cuales algunos son altamente determinantes para su funcionamiento. Esos parámetros, en optimización, son llamados variables de diseño, las cuales proporcionan información que permite describir el sistema u objeto a diseñar.

La elección de los parámetros importantes en un problema de optimización depende del diseñador en el momento del análisis del sistema u objeto. La eficiencia y velocidad de optimización depende del número de variables de diseño elegidas.

2.1.2. Restricciones

Las restricciones representan algunas relaciones funcionales entre las variables de diseño y otros parámetros satisfaciendo ciertos fenómenos físicos y ciertas limitaciones de recursos. Algunas de esas consideraciones requieren que el diseño permanezca en equilibrio ya sea dinámico o estático.

Existen, usualmente, dos tipos de restricciones: de igualdad y de desigualdad. Las restricciones de desigualdad establecen que las relaciones funcionales entre las variables de diseño son mayor que, menor que o iguales a cierto valor de un recurso.

Las restricciones de igualdad establecen las relaciones funcionales que deberían igualar exactamente el valor de un recurso. Las restricciones de igualdad son más difíciles de manejar y por lo tanto, necesitan ser evitadas en la medida de lo posible.

2.1.3. Función objetivo

La tercera tarea en el procedimiento de formulación es definir la función objetivo en términos de las variables de diseño y otros parámetros del problema. La función objetivo es una representación matemática donde se evalúan los valores de las variables de diseño para determinar si éstos son los óptimos. La función objetivo puede ser de dos tipos: de minimización o maximización. Por lo regular las técnicas de optimización son sólo para minimización o maximización, sin embargo si se desea cambiar el tipo de función objetivo se utiliza el principio de dualidad (Fig. 2.2) al multiplicar la función por -1.

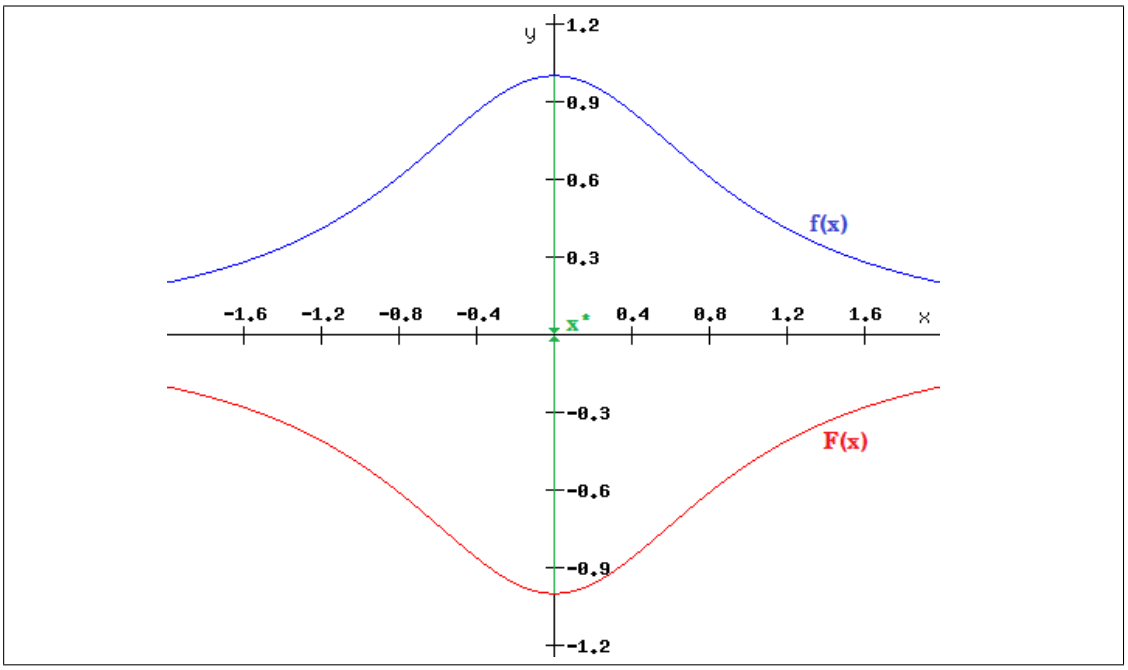

Figura 2.2: Principio de dualidad. El punto del máximo de $f(x)$ es el mismo que el mínimo de $F(x) = -f(x)$.

2.1.4. Límites de las variables

Los límites de las variables son los valores mínimos y máximos que una variable de decisión puede tomar, esto es:

$$x_i^{(L)} \leq x_i \leq x_i^{(U)} \text{ para } i = 1, 2, ..., N$$

donde x_i denota una variable de decisión, $x_i^{(L)}$ es el mínimo valor que puede tener y $x_i^{(U)}$ el máximo, N es el número de variables.

2.2. Optimización multiobjetivo

Cuando un problema de optimización involucra más de una función objetivo, la tarea de encontrar una o más soluciones óptimas es conocida como optimización multiobjetivo o multicriterio. Muchos problemas del mundo real son multiobjetivo, pues los objetivos a optimizar son igual de importantes. Diferentes soluciones pueden generar conflictos entre los objetivos, esto es, una solución mejora uno o varios objetivos pero empeora uno o varios otros objetivos.

Una de las diferencias principales con la optimización de un solo objetivo es que no existe una única solución sino un conjunto de ellas, debido a los conflictos entre los objetivos [16].

En los problemas de optimización multiobjetivo existen tres posibles situaciones:

- Minimizar todas las funciones objetivo
- Maximizar todas las funciones objetivo
- Minimizar algunas y maximizar otras

Por razones de simplicidad se convierten todas ya sea a minimización o maximización, por el principio de dualidad visto anteriormente.

En las siguientes subsecciones se mostrarán las definiciones de un problema de optimización multiobjetivo, óptimo de Pareto, dominancia de Pareto y Frente de Pareto, encontradas en [9].

2.2.1. Definición formal

Un problema de optimización multiobjetivo se define formalmente como:

Encontrar el vector de variables de decisión $\vec{x}^* = [x*_1, x*_2, ..., x*_n]^T$ el cual satisface las m restricciones de desigualdad:

$$g_i(\vec{x}) \geq 0 \ i = 1, 2, ..., m$$

las p restricciones de igualdad

$$h_i(\vec{x}) = 0 \ i = 1, 2, ..., p$$

y optimiza el vector de funciones

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]^T$$

Las restricciones de desigualdad e igualdad definen la zona factible Ω y cualquier punto $\vec{x}$ en Ω define una solución factible.

2.2.2. Óptimo de Pareto

Ya que en optimización multiobjetivo se tienen diversas objetivos por optimizar, la noción de "óptimo" cambia, ésto es porque la finalidad es encontrar buenos compromisos, en lugar de una única solución. La definición de "óptimo"que ha sido más aceptada es la propuesta por Francis Ysidro Edgeworth en 1881 y posteriormente generalizada por Vilfrido Pareto en 1896 la cual se le conoce como óptimo de Pareto.

La definición formal de óptimo de Pareto es la siguiente:

Un vector de variables de decisión $\vec{x}^* \in \Omega$ es óptimo de Pareto si para cada $\vec{x} \in \Omega$ y $I = 1, 2, ..., k$

$$\forall i \in I(f_i(\vec{x}) \geq f_i(\vec{x}^*))$$

y existe al menos una $i \in I$ tal que

$$f_i(\vec{x}) > f_i(\vec{x}^*)$$

Ésto quiere decir que un vector de variables de decisión que pertenece a la zona factible se le considera como óptimo de Pareto si al compararlo con los demás vectores de variables de decisión que pertenecen a la zona factible cumple con dos condiciones: la primera es que el valor de sus funciones objetivo sean menores o iguales a los valores de los otros vectores. La segunda es que, al menos una función objetivo, sea menor.

Una explicación gráfica del concepto de óptimo de Pareto puede verse en la Fig. 2.3.

Al conjunto de óptimos de Pareto se le denomina P^* y se define como:

$$P^* = \vec{x}^* \in \Omega$$

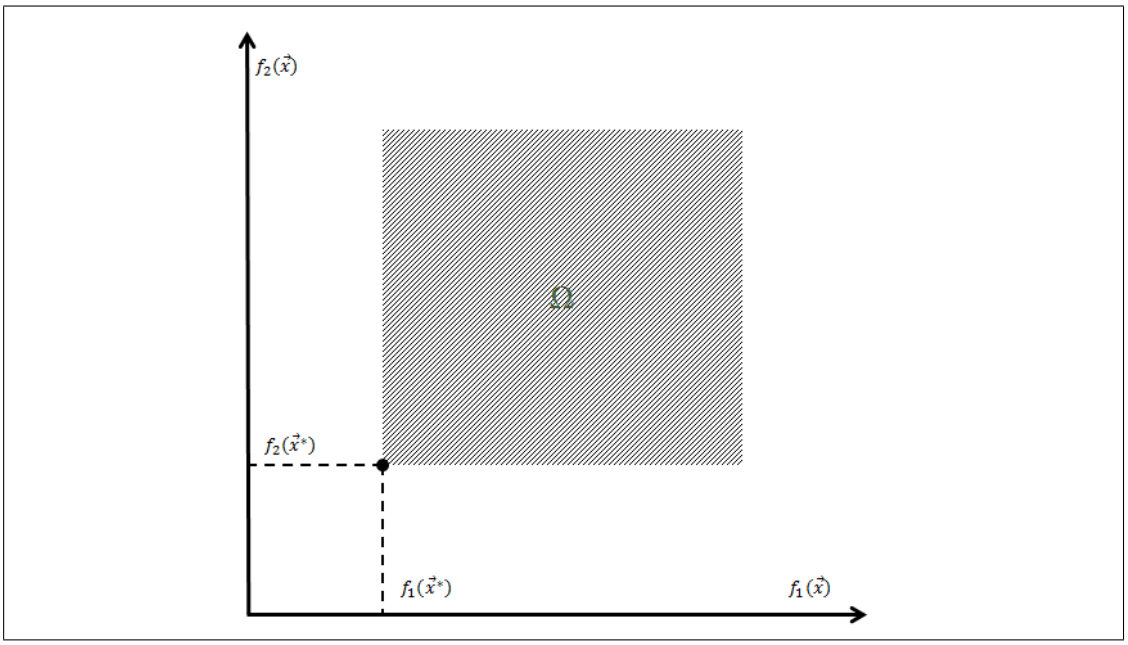

Figura 2.3: Único óptimo de Pareto de la región factible, suponiendo minimización. (Tomado de [9])

2.2.3. Dominancia de Pareto

En la resolución de problemas de optimización multiobjetivo es necesario identificar a aquellas soluciones que son dominadas por otras para eliminarlas del conjunto de soluciones. El concepto de dominancia es el siguiente:

Un vector de funciones objetivo $\vec{u} = [u_1, u_2, ..., u_k]$ se dice que domina a otro vector de funciones objetivo $\vec{v} = [v_1, v_2, ..., v_k] (\vec{u} \preceq \vec{v})$ si y solo si $\vec{u}$ es parcialmente menor a $\vec{v}$, es decir, $\forall i \in 1, ..., k, u_i \leq v_i \wedge \exists i \in 1, ...k : u_i < v_i$

Esto es, una solución $\vec{u}$ domina a otra solución $\vec{v}$ si:

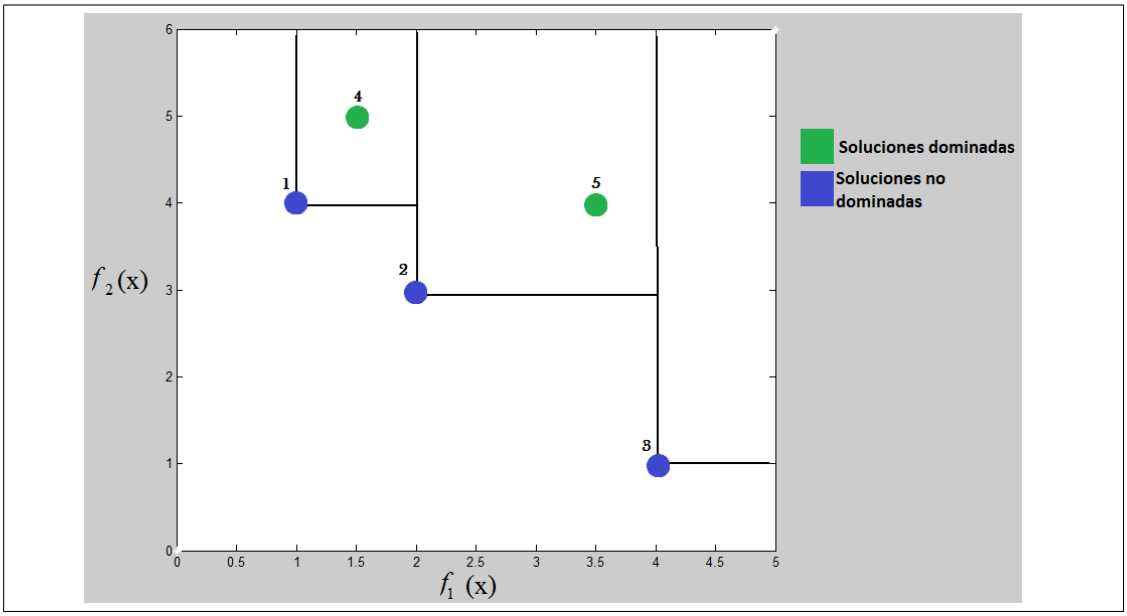

Figura 2.4: Ejemplo de soluciones dominadas y no dominadas, asumiendo minimización

- La solución $\vec{u}$ es mejor que la solución $\vec{v}$ en todos sus objetivos.
- La solución $\vec{u}$ es mejor que la solución $\vec{v}$ en al menos un objetivo y es igualmente buena en los demás objetivos.

Entre dos soluciones existen tres posibilidades:

1. $\vec{u}$ domina a $\vec{v}$.
2. $\vec{u}$ es dominada por $\vec{v}$.
3. $\vec{u}$ y $\vec{v}$ son no dominadas entre sí.

Como se observa en la Fig. 2.4 los puntos en azul son las soluciones no dominadas, mientras que los puntos en verde son las soluciones dominadas. El punto 1 domina al 4 pues los valores de f_1 y f_2 del punto 1 son menores que f_1 y f_2 del punto 4. Al comparar el punto 3 con el 5 se observa que el punto 3 sólo es menor en f_2, por lo tanto, no se puede decir que el punto 3 domine al 5, sin embargo, el punto 2 domina al 5 pues sus valores de f_1 y f_2 son menores.

2.2.4. Frente de Pareto

Dado un problema multiobjetivo $\vec{f}(x)$ y un conjunto de óptimos de Pareto P^*, el frente de Pareto (PF^*) es definido como:

$$PF^* := \{\vec{u} = \vec{f} = (f_1(x), ..., f_k(x)|x \in P^*)\}$$

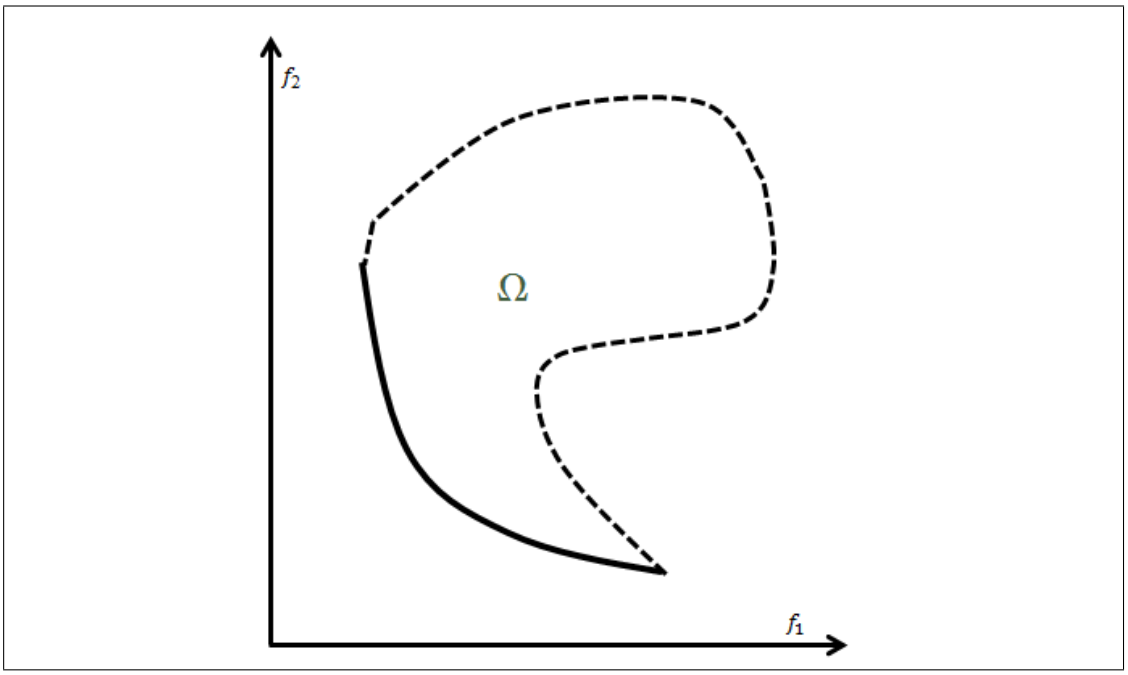

Figura 2.5: Frente de Pareto (línea continua) en un problema de dos objetivos

El frente de Pareto PF^* es el conjunto de vectores de funciones objetivo cuyos vectores de variables de decisión pertenecen al conjunto de óptimos de Pareto. Es un conjunto de soluciones no dominadas.

Es muy difícil, en la mayoría de los casos imposible, encontrar una expresión analítica de la línea o superficie que contiene estos puntos. Para generar un frente de Pareto el procedimiento normal es computar el conjunto de puntos en Ω y su correspondiente $f(\vec{x} \in \Omega)$. Cuando hay un número suficiente de ellos entonces se determinan los puntos no dominados y es posible generar una aproximación al frente de Pareto.

En un problema de optimización multiobjetivo se requieren encontrar soluciones que se encuentren el frente óptimo de Pareto y que éstas estén bien distribuidas en él [16].

En la Fig. 2.5 se puede observar gráficamente un frente de Pareto de un problema de dos funciones objetivo.

Capítulo 3

Cadena de suministro

Una cadena de suministro es un conjunto de actividades funcionales (transporte, control de inventarios, etc.) que se repite muchas veces a lo largo del canal de flujo, mediante los cuales la materia prima se convierte en producto terminado y se añade valor para el consumidor [1].

Esta definición nos dice que una cadena de suministro consiste en un canal de flujo a través del cual se transforma la materia prima en un producto terminado. Este flujo consta de varias etapas o escalones donde se le añade valor al producto, dependiendo de la actividad de cierto escalón. Un ejemplo de cadena de suministro lo podemos observar en la Fig. 3.1, en la cual se observan los diferentes escalones o etapas que un producto recorre hasta llegar al cliente final; comenzando desde la obtención de materias primas, posteriormente la transformación de éstas en materiales necesarios para la fabricación del producto, la manufactura, distribución y venta del producto cuyo destino final es el consumidor.

Una cadena de suministro presenta diferentes retos, entre los principales se encuentran: planear una estrategia de administración de recursos y satisfacción de demandas, seleccionar proveedores que ofrecerán los recursos y servicios requeridos en la elaboración de un producto, manufacturar el producto, entregar el producto a los clientes y realizar acuerdo de devolución de producto en caso de existir un fallo.

El desempeño de una cadena de suministros de manera general se observa haciendo un balance entre costos de operación y las ganancias obtenidas, aunque también existen otras medidas de desempeño usadas dentro del modelado de cadenas de suministro. Por ejemplo, en [4] se describen las más comunes y que tratan de minimizarse, como son: costos de operación, niveles medios de inventario, cantidad de inventario obsoleto, probabilidad de agotamiento de inventario, varianza de la demanda de producto o amplificación de la demanda, número de días de actividad, entre otras. Entre las medidas de desempeño que se quieren maximizar encontramos: ganancias, beneficios del comprador - proveedor y la capacidad del sistema disponible.

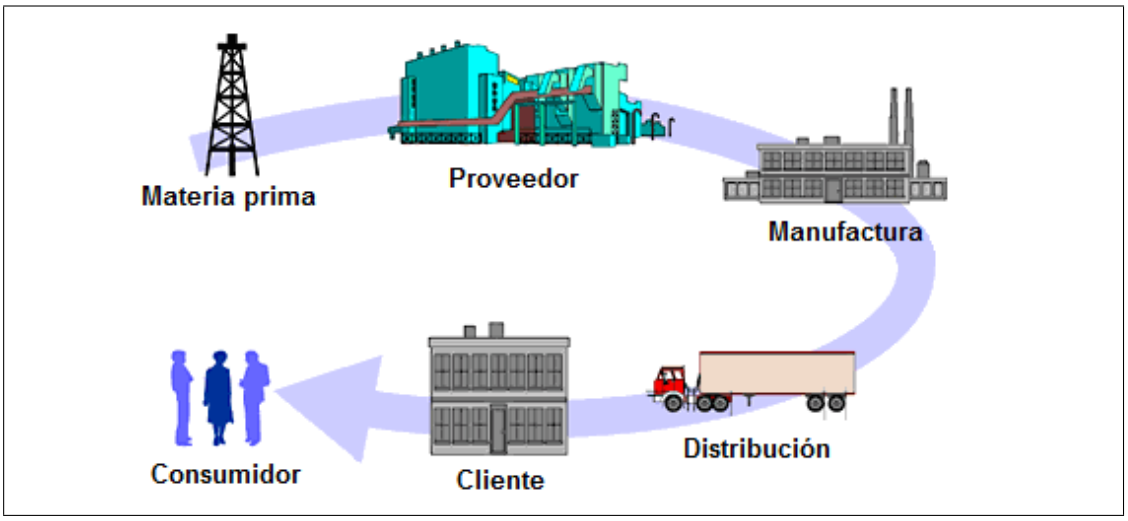

Figura 3.1: Ejemplo de cadena de suministro [42].

Se puede realizar un modelo matemático de una cadena de suministro y optimizarlo para cumplir ciertos objetivos deseados, sin embargo, estos modelos pudieran ser difíciles de resolver mediante métodos tradicionales de búsqueda tales como programación lineal, diferenciación o métodos basados en gradiente. Es por eso que se utilizan técnicas meta-heurísticas para determinar buenas soluciones [11].

Otro factor que dificulta el encontrar una solución óptima en una cadena de suministro es que sus objetivos pueden estar en conflicto, por ejemplo, se quiere mantener costos mínimos de inventario pero a la vez estar preparados para la demanda del cliente y maximizar la satisfacción de éste, si se mantienen niveles de inventario bajos los costos se minimizan pero tal vez no se satisfaga la demanda del cliente o bien si se mantienen inventarios altos el cliente siempre estará satisfecho pero los costos de mantenimiento aumentan.

3.1. Efecto látigo

Como se mencionó anteriormente, las industrias para mejorar el desempeño de su producción y maximizar sus ganancias, intentan eliminar los problemas generados en las operaciones de sus cadenas de suministro.

Algunos de esos problemas son un exceso de inventario, falta de producto disponible, distorsión en la información que describe la operación de la cadena de suministro y transporte de mercancía insuficiente.

Uno de estos problemas es el llamado efecto látigo. El efecto látigo es la amplificación de la varianza de la demanda mientras se mueve a lo largo de los escalones superiores desde los escalones inferiores [29], ver la Fig. 3.2.

En los años 60 el profesor Jay Forrester del MIT, analizó un fenómeno que se presentaba en la cadena de suministro de P&G [22]. Observó que un pequeño

movimiento de la demanda en un punto de venta acababa produciendo un gran movimiento en los escalones sucesivos de la cadena, cada vez mayor conforme subía.

En 1989 en el MIT, se desarrolló un modelo computacional de simulación para estudiar el efecto látigo al cual se le llamó el juego de la cerveza. El juego de la cerveza consiste en cuatro escalones: cliente, vendedor, distribuidor y fábrica. Cada escalón realiza su pedido al escalón superior de acuerdo a cierta política de orden. Del juego se infieren tres consecuencias:

1. Aparecen largas oscilaciones en órdenes e inventarios.

2. La demanda se incrementa en los escalones más altos.

3. La tasa de órdenes tiende a subir desde el vendedor hasta la fábrica.

Estas fluctuaciones a lo largo de la cadena son producto del efecto látigo y crean mucha inestabilidad además de que vuelven ineficientes las operaciones, aumentan los costos de producción y mano de obra o no se cumple con la demanda del cliente [21].

Algunas de las causas del efecto látigo que se han identificado son las siguientes[23]:

- Actualización del pronóstico de demanda: los diferentes escalones de la cadena de suministro basan sus expectativas de demanda con base en un histórico de órdenes recibidas. Un incremento en la demanda lleva a un aumento en la expectativa del tamaño de la orden, la cual es transferida al siguiente escalón, incrementando así y cada vez más, el tamaño de las órdenes. Este comportamiento funciona también a la inversa, cuando la demanda decrementa el tamaño de la orden esperado disminuye.

- Órdenes por lotes: cuando las demandas llegan, éstas impactan en el inventario, pero los actores de una cadena de suministro no siempre la atienden inmediatamente, pues puede que estén esperando a acumular cierto tamaño de orden, ya sea por los costos fijos de ésta o por eficiencia en la distribución. En esta situación puede observarse un aumento en la demanda para cubrir el tiempo que tarda en llegar el pedido.

- Fluctuaciones en los precios: cuando los precios suben o bajan, los clientes tienden a decrementar o incrementar, respectivamente, su demanda, creando con ello una inestabilidad en los niveles de inventario cuando los precios se normalizan.

- Racionamiento: cuando la demanda de algún producto supera la oferta, el proveedor puede racionar el producto a sus clientes, sabiendo ésto, los clientes harán un mayor pedido del que realmente necesitan. Cuando ya se puede cumplir con la demanda, los clientes no ordenan y los inventarios se quedan con niveles altos.

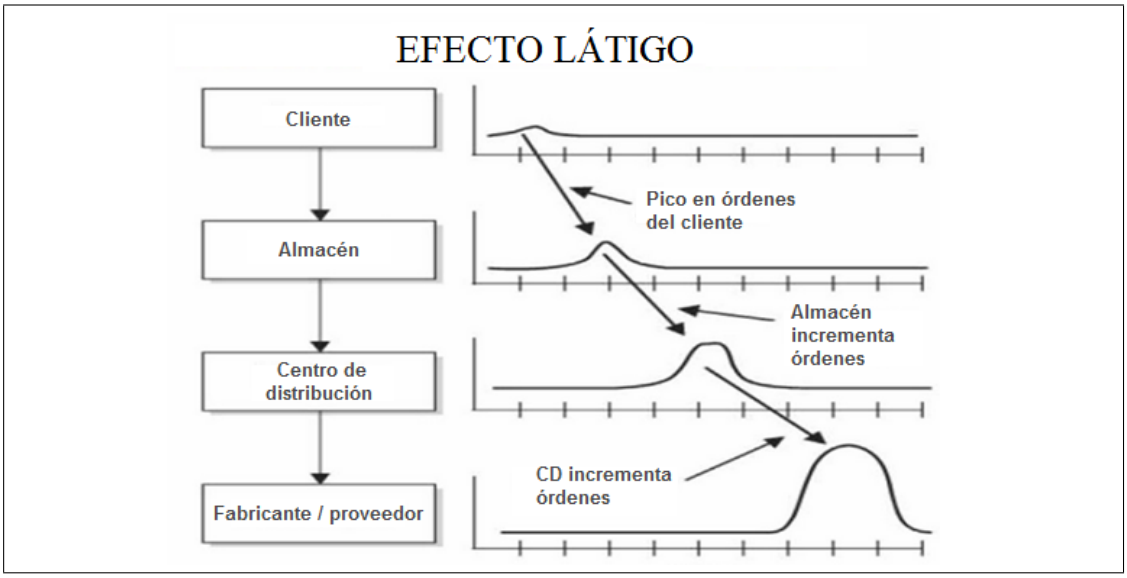

Figura 3.2: Efecto látigo [32].

Se han hecho diversos estudios para tratar de mitigar este efecto. Las técnicas de reducción de efecto látigo se clasifican en varias categorías[21] y son las siguientes:

1. Control de sistemas: fortalece la estabilidad dinámica de la cadena de suministro.

2. Compresión del tiempo: se enfocan en reducir material y tiempo de despacho de órdenes.

3. Transparencia de información: se comparte información entre miembros de la cadena.

4. Eliminación de escalones: se remueven escalones.

Por la naturaleza multidiversa de las cadenas de suministro, no siempre se pueden aplicar medidas de reducción que contemplen todas las categorías, es por eso que se buscan nuevas técnicas que hagan eficiente la operación de las cadenas.

3.2. Optimización en cadenas de suministro

Son múltiples los trabajos que se han realizado en torno a la problemática de optimizar una cadena de suministro. En [7] se describe un modelo de cadena de suministro, el cual cuantifica las ganancias totales y los costos relacionados con la calidad de los productos. Algunos otros trabajos han sido orientados al análisis del efecto látigo, como es el caso de [21] en el cual se mencionan algunas causas

encontradas que lo generan y trabajos que se han ido realizando con el objetivo de eliminarlo. Mismo caso que en [50], donde también se mencionan algunas causas que lo provocan así como también técnicas que se utilizan para reducirlo: estableciendo un mecanismo para compartir información; estableciendo una alianza estratégica, fortaleciendo la cooperación verdadera; control de inventario VMI (Inventario Manejado por Vendedor), unión de control de inventario, reducción del tiempo de manejo de la cadena.

En [48] se busca reducir el efecto látigo y los costos totales utilizando una técnica de control basada en la divergencia de un sistema. En [39] se habla del uso de especuladores para mitigar el efecto látigo, la técnica se probó mediante una simulación basada en agentes y se observó que el efecto látigo se reducía, sin embargo los costos totales de operación en la cadena de suministro aumentaban. La técnica de compartir información fue analizada en [38]. En [13] se usa una solución basada en un sistema multiagente. Se han utilizado también algoritmos genéticos para reducir el efecto látigo como es el caso de [40] y de [43] trabajo en el cuál se utiliza un algoritmo basado en evolución gramatical.

Existen también diversos ejemplos de trabajos en los cuales se busca optimizar más de un objetivo en una cadena de suministro. En trabajos recientes se ha incluido como objetivo reducir el impacto ambiental de las operaciones realizadas en una cadena de suministro además de reducir costos de operación, [45] y [44] son ejemplos de este tipo de trabajos, en ambos se utilizaron soluciones basadas en lógica difusa, [44] además presenta un caso de estudio realizado en Portugal donde fue probada la solución. En [34] se utilizó un algoritmo basado en el comportamiento forrajero de las abejas para minimizar los costos totales y el plazo de entrega en una cadena de suministro. En [8] se buscaron minimizar los costos totales y el tiempo de entrega y maximizar la calidad, se utilizaron dos algoritmos Pareto Genetic Algorithm y Modified Pareto Algorithm, el modelo optimizado estaba basado en una cadena del tipo Build to Order. Minimizar los costos totales y maximizar el servicio al cliente fueron los objetivos a optimizar en [31], trabajo en el cual se utilizó el popular algoritmo evolutivo multiobjetivo llamado NSGA-II.

En [6] se mencionan algunas prácticas y técnicas que se utilizan para reducir el efecto látigo y minimizar los costos totales de operación de una cadena de suministro, mismos objetivos perseguidos en [49] donde se utilizó una técnica de control basada en la divergencia de un sistema.

Recientemente en [3] se propuso el uso de meta-heurísticas para reducir el efecto látigo y los costos totales de operación, para tal fin se utilizó NSGA-II con una modificación en su operador de mutación para optimizar estos dos objetivos. El modelo formal abordado en esta tesis está basado en el mostrado en dicho trabajo.

3.3. Modelo formal de minimización de costos totales y efecto látigo

El modelo del problema a abordar es una adaptación del descrito en [2], el cual describe la interacción de dos escalones de una cadena de suministros. El modelo consiste en D distribuidores i, los cuales envían P productos p, a los W mayoristas j, durante T unidades de tiempo t.

El modelo formal del problema abordado es el siguiente:

Minimizar:

$$TC = \sum_{i=1}^{D}\sum_{j=1}^{W}\sum_{p=1}^{P} c'^{p} U_{DWijp} + \sum_{i=1}^{D}\sum_{p=1}^{P} c^{ip} I_i^p + \sum_{j=1}^{W}\sum_{p=1}^{P} c^{jp} I_j^p \tag{3.1}$$

$$BWE = \sum_{p=1}^{P}\sum_{j=1}^{W} \frac{Var(\sum_{i=1}^{D} q_{ijt}^p)}{Var(dm_{jt}^p)} \tag{3.2}$$

Sujeto a las restricciones

$$\sum_{p=1}^{P}\sum_{j=1}^{W} U_{DWijp}(t) + \sum_{p=1}^{P} I_i^p(t) \leq F_{Di} \tag{3.3}$$

$$\sum_{p=1}^{P} I_j^p(t) \leq F_{wj} \tag{3.4}$$

$$I_i^p(t) + \sum_{j=1}^{W} U_{DWijp}(t) \geq BS_i^p \tag{3.5}$$

$$U_{DWijp}(t) \leq q_{ijt}^p \tag{3.6}$$

$$U_{min}^p \leq U_{DWijp}(t) \leq U_{max}^p \tag{3.7}$$

$$BS_i^p \leq I_i^p(t) \leq I_{i-max}^p \tag{3.8}$$

$$BS_j^p \leq I_j^p(t) \leq I_{j-max}^p \tag{3.9}$$

$$q_{ijt-min} \leq q_{ijt}^p \leq q_{ijt-max} \tag{3.10}$$

La definición de las variables y parámetros se encuentra en la Tabla 3.1

Variables de decisión:

U_{DWijp}	Número de unidades de producto p transportados de un distribuidor i a un mayorista j
q^p_{ijt}	Cantidad de producto p ordenado por un mayorista j a un distribuidor i en el tiempo t es igual a $I^p_j(t) - I^p_j(t-1) + dm^p_{j(t-1)}$ (política de orden)
I^p_i	Inventario de producto p en el distribuidor i
I^p_j	Inventario de producto p en el mayorista j

Parámetros:

P	Número total de productos
D	Número total de distribuidores
W	Número total de mayoristas
i	Índice para el número de distribuidores
j	Índice para el número de mayoristas
p	Índice para el número de productos
c'^p	Costo por unidades transportadas de un producto p
c^{ip}	Costo de mantenimiento de inventario del distribuidor i de un producto p
c^{jp}	Costo de mantenimiento de inventario del mayorista j de un producto p
dm^p_{jt}	Demanda de producto p del mayorista j en el tiempo t
F_{Di}	Capacidad fijada del distribuidor i
F_{Wj}	Capacidad fijada del mayorista j
U^p_{min}	Mínimas unidades transportadas del producto p
U^p_{max}	Máximas unidades transportadas del producto p
I^p_{imax}	Máximas unidades de inventario del producto p en el distribuidor i
I^p_{jmax}	Máximas unidades de inventario del producto p en el mayorista j
q^p_{ijtmin}	Mínima cantidad de orden de un producto p de un mayorista j al distribuidor i
q^p_{ijtmax}	Máxima cantidad de orden de un producto p de un mayorista j al distribuidor i
BS^p_i	Buffer stock de un producto p en el distribuidor i
BS^p_j	Buffer stock de un producto p en el mayorista j

Tabla 3.1: Variables y parámetros del modelo modificado

P	2		D	1
W	2		T	10
c'^p	2.5		c^{ip}	3.0
c^{jp}	2.0		F_{Di}	1000
F_{Wj}	900,1000		U^p_{min}	3
U^p_{max}	300		I^p_{imax}	500,580
I^p_{jmax}	300,280		q^p_{ijtmin}	5
q^p_{ijtmax}	400		BS^p_i	15
BS^p_j	10			

Tabla 3.2: Valores de los parámetros

En este trabajo se consideraron valores de $D = 1$, $W = 2$, $P = 2$ y $T = 10$ lo cual nos da un total de 100 variables y 50 restricciones.

El valor de los parámetros se encuentra en la Tabla 3.2

Las variables del modelo (unidades transportadas, inventario en el distribuidor e inventario en el mayorista) son todas de tipo entero.

El objetivo mostrado en la Ec. 3.1 representa los costos totales, los cuales consisten de costos de transportación de las unidades (primer término) y los costos de mantenimiento de inventario del distribuidor y del mayorista (últimos dos términos). El objetivo de la Ec. 3.2 representa el efecto látigo el cual se calcula dividiendo la varianza de la cantidad de órdenes sobre la varianza de la cantidad demandada.

La restricción presentada en la Ec. 3.3 establece que la suma de unidades totales transportadas para los productos de un distribuidor a los mayoristas debe estar de acuerdo al límite de la capacidad fijada de dicho distribuidor. La restricción de la Ec. 3.4 se asegura que el inventario total de los productos en un mayorista deben estar de acuerdo al límite de la capacidad fijada de ese mayorista. Por otro lado, la restricción de la Ec. 3.5 asegura que la suma del inventario de un producto p en un distribuidor i y las unidades transportadas de este distribuidor a los mayoristas deberán ser mayores que el buffer stock de un producto p en dicho distribuidor. La restricción de la Ec. 3.6 asegura que las unidades transportadas de un distribuidor i al mayorista j para un producto p debe ser menor que la cantidad de producto p ordenada por el mayorista j al distribuidor i. Finalmente, las restricciones de las Ecs. 3.7 a 3.10 establecen los limites superiores e inferiores de las variables usadas en la formulación.

Capítulo 4

MOMBFOA

4.1. Algoritmos de optimización basados en inteligencia colectiva

La Inteligencia Colectiva es un tipo de algoritmo meta-heurístico inspirado en el comportamiento colaborativo y asociativo de animales tales como colonia de hormigas, abejas, parvadas de aves, bacterias, entre otros, los cuales se caracterizan por formar grupos y trabajar en conjunto para alcanzar un objetivo [27].

En estos grupos cada individuo se comunica con los demás, ya sea directa o indirectamente, actuando en su ambiente local. Al comportamiento que tienen los grupos al interactuar para resolver problemas se le conoce como Inteligencia Colectiva. A los modelos algorítmicos basados en dicho comportamiento se les denomina Inteligencia Colectiva Computacional.

El objetivo de la Inteligencia Colectiva Computacional es modelar el comportamiento simple de individuos y las interacciones locales de éstos con el ambiente e individuos vecinos, para obtener comportamientos más complejos que sean usados para la resolución de problemas complejos, principalmente problemas de optimización [20].

Los algoritmos de Inteligencia Colectiva trabajan de la siguiente manera[25]:

1. Generan un conjunto (población) de soluciones (individuos) al problema.
2. Evalúan cada individuo en la función objetivo a optimizar.
3. Seleccionan a los mejores individuos de la población con base en su valor en la función objetivo.
4. Generan nuevas soluciones a partir de las mejores soluciones utilizando operadores de variación.
5. Evalúan las nuevas soluciones.

6. Escogen las soluciones que formarán parte de la siguiente iteración (generación).

Para la representación de las soluciones se utilizan, por lo general, números reales, la selección consiste en escoger a los individuos que tengan un mejor valor en la función objetivo. Los operadores de variación están basados en procesos de comportamientos como reproducción, variaciones aleatorias y movimientos cooperativos. El reemplazo consiste en seleccionar a los individuos que pasarán a la siguiente generación. En los algoritmos de Inteligencia Colectiva el tamaño de la población permanece constante en cada generación.

En esta sección de la tesis se hablará sobre un algoritmo de Inteligencia Colectiva basado en el comportamiento forrajero de las bacterias E. Coli, el BFOA [41].

4.2. BFOA

El BFOA (Bacterial Foraging Optimization Algorithm) fue propuesto en [41] en 2002. Está inspirado en el comportamiento social y cooperativo de la bacteria E. Coli cuando está en busca de nutrientes. Cada bacteria trata de maximizar la energía obtenida por unidad de tiempo a la vez de anular sustancias nocivas, las bacterias se comunican entre sí. Una población de bacterias se comporta de la siguiente manera:

1. Las bacterias están distribuidas aleatoriamente sobre un mapa de nutrientes.

2. Las bacterias se mueven hacia regiones con alta concentración de nutrientes en el mapa por medio de quimiotaxis (giro-nado). Aquellas localizadas en regiones nocivas o con bajos nutrientes mueren y se dispersan. Aquellas que están en regiones altas en nutrientes se replicarán.

3. Las bacterias están ahora localizadas en regiones prometedoras del mapa de nutrientes entonces tratan de atraer otras bacterias generando atractores químicos.

4. Las bacterias están localizadas ahora en las regiones con más altos niveles de nutrientes.

5. Las bacterias se dispersan para buscar nuevas regiones de nutrientes en el mapa.

Basado en ese comportamiento de las bacterias, Passino propuso el algoritmo de optimización de bacterias en búsqueda de alimento, el pseudocódigo se observa en la Fig. 4.1.

```
1  Inicialización de parámetros
2  Crear una población inicial aleatoria de bacterias θ^i(j,k,l)∀i, i = 1,...,S_b
3  Evaluar f_k(θ^i(j,k,l)) ∀i, i = 1,...,S_b
4  for l = 1 to N_ed do
5      for k = 1 to N_re do
6          for j = 1 to N_c do
7              for i = 1 to S_b do
8                  Llevar a cabo el proceso quimiotáxico para la bacteria
                   θ^i(j,k,l) los nados son limitados porN_s
9              end
10         end
11         Realizar el paso de reproducción para la eliminación de S_r bacterias
           y duplicar la otra mitad
12     end
13     Realizar el paso de eliminación-dispersión para todas las bacterias
       θ^i(j,k,l)∀i, i = 1,...,S_b con probabilidad 0 ≤ P_ed ≤ 1
14 end
```

Figura 4.1: BFOA original. Los parámetros de entrada son el número de bacterias S_b, límite del paso quimiotáxico N_c, límite del paso de nado N_s, límite del ciclo de reproducción N_{re}, número de bacterias a reproducir S_r, límite del ciclo de eliminación-dispersión N_{ed}, tamaño de paso C_i y probabilidad de eliminación-dispersión P_{ed}.

Cada bacteria i representa una solución potencial al problema de optimización y se define como $\theta^i(j,k,l)$ donde j es el ciclo quimiotáxico, k es el ciclo de reproducción y l el ciclo de eliminación-dispersión. El giro representa una dirección aleatoria $\phi(i)$ definida como:

$$\phi(i) = \frac{\Delta(i)}{\sqrt{\Delta(i)^T \Delta(i)}}$$

donde $\Delta(i)$ es un vector generado aleatoriamente de tamaño n con sus elementos dentro del intervalo $[-1, 1]$. Después de modificar su posición se lleva a cabo el ciclo de nado:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(i)$$

donde $C(i)$ es el tamaño de paso definido por el usuario.

El nado se repite N_s veces si y solo si la nueva posición es mejor que la previa, de otra forma un nuevo giro se realiza. El proceso de reproducción consiste en ordenar todas las bacterias de la población basado en el valor de su función objetivo y eliminar la peor mitad, la mitad restante se duplica. El proceso de eliminación-dispersión consiste en eliminar cada bacteria con cierta probabilidad $0 \leq P_{ed} \leq 1$.

Siguiendo los pasos del pseudocódigo de la Fig.4.1 el proceso de optimización llevado a cabo por BFOA es el siguiente:

1. Como primer paso se inicializan los parámetros.

2. Se procede a generar una población aleatoria de bacterias $\theta^i(j,k,l) \forall i, i = 1, ..., S_b$, el tamaño de la población debe ser constante durante todo el proceso del algoritmo.

3. Posteriormente se evalúan cada una de las bacterias $f(\theta^i(j,k,l)) \forall i, i = 1, ..., S_b$, es decir se sustituye en la función objetivo del problema a resolver los valores de las variables de la bacteria en proceso, ésto se realiza para toda la población de bacterias.

4. Se incrementa en 1 el ciclo de eliminación-dispersión, reproducción y quimiotáxico, $N_{ed}+1, N_{re}+1, N_c+1$ hasta el número máximo establecido en la iniciación de parámetros.

5. Posteriormente cada una de las bacterias realizan su primer ciclo quimiotáxico, es decir, realizan su búsqueda, nadando y avanzando hacia una nueva posición, los nados de cada una de las bacterias son sólo el número permitido por N_s la elección de un nado se da cuando se mejoró el valor de la función objetivo en la nueva posición. De otra manera se realizará un giro.

6. A continuación se realiza el proceso de reproducción, en el cual se reproducen las S_r (mitad) mejores bacterias y se elimina a la otra mitad.

7. Finalmente se realiza el proceso de eliminación-dispersión de bacterias de acuerdo a la probabilidad determinada por P_{ed}.

8. Se incrementan los ciclos, eliminación-dispersión, reproducción y quimiotáxico y se vuelven a realizar lo descrito en los puntos 5, 6 y 7 hasta cumplir con el número máximo de ciclos establecidos en la iniciación de parámetros.

4.3. MBFOA

El MBFOA (Modified Bacterial Foraging Optimization Algorithm) es una versión modificada del BFOA propuesta en [35]. Se simplificó el algoritmo y se eliminaron algunos parámetros para adaptarlo a la resolución de problemas en ingeniería.

Las cuatro modificaciones hechas en MBFOA son las siguientes:

1. Un solo ciclo para incluir el ciclo quimiotáxico, la reproducción y eliminación-dispersión.

   El ciclo de generaciones es controlado por el parámetro $GMAX$ en el cual las bacterias realizan su ciclo quimiotáxico. Después de ésto se lleva a cabo un solo paso de reproducción y un solo paso de eliminación-dispersión dentro de un mismo ciclo de generación. En el paso de eliminación-dispersión sólo se elimina a la peor bacteria de la población.

2. El ciclo quimiotáxico se lleva a cabo mediante movimientos de giro-nado de la bacteria del mismo modo que en BFOA, pero el parámetro $C(i)$ ya no lo proporciona el usuario sino que se calcula mediante:

   $$C(i)_k = R^*(\tfrac{\Delta\vec{x}_k}{\sqrt{n}}), k = 1, ..., n$$

   donde $\Delta\vec{x}_k$ es la diferencia entre el límite superior (U_k) y el límite inferior (L_k) de las variables de diseño, n es el número de variables de diseño y R se usa para definir un porcentaje del valor usado por una bacteria como tamaño de paso.

   Si al moverse alguna bacteria queda fuera del límite de sus variables de diseño, se aplica:

   $$x_i = \begin{cases} 2L_i - x_i & \text{si } x_i < L_i \\ 2U_i - x_i & \text{si } x_i > U_i \\ x_i & \text{otro caso} \end{cases} \tag{4.1}$$

3. Se le agregó un mecanismo para el manejo de restricciones que modifica el criterio de selección que se basa en 3 reglas [15]:

```
Inicialización de parámetros
Crear una población inicial aleatoria de bacterias θ^i(j, G)∀i, i = 1, ..., S_b
Evaluar f(θ^i(j, G)) ∀i, i = 1, ..., S_b
for G = 1 to GMAX do
    for i = 1 to S_b do
        for j = 1 to N_c do
            Realizar el paso quimiotáxico para la bacteria θ^i(j, G) usando las
            reglas de factibilidad y las ecuaciones de nado y atracción.
        end
    end
    Realizar el paso de reproducción para la eliminación de las S_re (mitad)
    peores bacterias y duplicar la otra mitad, basado en las reglas de
    factibilidad.
    Eliminar la peor bacteria θ^w(j, G) de la población, basado en las reglas
    de factibilidad y generar una nueva aleatoriamente.
end
```

Figura 4.2: MBFOA. Los parámetros de entrada son el número de bacterias S_b, límite del paso quimiotáxico N_c, número de bacterias a reproducirse S_{re}, factor de escalamiento β, porcentaje del tamaño de paso R y el número de generaciones $GMAX$.

- Entre dos bacterias factibles la del mejor valor de la función objetivo es preferida.
- Entre una bacteria factible y una no factible se prefiere la factible.
- Entre dos bacterias no factibles la que tenga menor suma de violación de restricciones es preferida.

4. En el movimiento de atracción, todas las bacterias de la población se mueven hacia la mejor de todas, lo cual se hace mediante:

$$\theta^i(j+1, G) = \theta^i(j, G) + \beta(\theta^B(G) - \theta^i(j, G))$$

donde $\theta^B(G)$ es la posición actual de la mejor bacteria de la población en la generación G y β define qué tanto se acercarán las bacterias a la mejor. El movimiento de atracción se aplica dos veces en un ciclo quimiotáxico, mientras en los pasos restantes se aplica el operador de giro-nado.

El pseudocódigo de MBFOA se observa en la Fig. 4.2.

4.4. MOMBFOA

MOMBFOA (Multi-Objective Modified Bacterial Foraging Optimization Algorithm) fue propuesto en [36] como una adaptación del MBFOA destinada a la resolución de problemas multiobjetivo con restricciones.

Se cambiaron los siguientes mecanismos de MBFOA: criterio de selección, uso de un archivo externo como mecanismo de elitismo y manejo de diversidad.

4.4.1. Dominancia de Pareto y mecanismo de manejo de restricciones

En MOMBFOA el criterio de selección está basado en la dominancia de Pareto y en las reglas de factibilidad para el manejo de restricciones, de ello se tienen tres reglas de selección que son las siguientes:

- Entre dos bacterias factibles, la que domine a otra es preferida. Si ambas son factibles y no dominadas, se escoge una aleatoriamente.
- Entre una bacteria factible y una no factible se prefiere la factible.
- Entre dos bacterias no factibles se prefiere la que tenga menor suma de violación de restricciones.

4.4.2. Archivo externo

MOMBFOA utiliza un archivo (conjunto externo) para guardar solo las soluciones factibles no dominadas. El archivo, el cual al principio está vacío, se actualiza en cada ciclo del algoritmo insertando una copia de las bacterias factibles de la población después del ciclo quimiotáxico. Cada vez que un conjunto de bacterias entra al archivo se verifica que se conserven únicamente las no dominadas.

4.4.3. Mecanismo de diversidad

Con el fin de mantener adecuadamente la diversidad de las soluciones en el espacio de las funciones objetivo se consideran mecanismos adicionales en el algoritmo. MOMBFOA utiliza la distancia crowding, la cual consiste en asignar un valor de cercanía con otras soluciones. Mientras más aislada esté una solución mayor será su distancia crowding. Las bacterias con más alto valor de distancia crowding son preferidas. El pseudocódigo y una descripción gráfica de la distancia crowding se encuentran en las Figs. 4.3 y 4.4.

input: Conjunto de soluciones I

$l = |I|$; // Número de soluciones en I

foreach i **do**

  $I[i]_{distancia} = 0$; // Se inicializan distancias

end

foreach *objetivo* m **do**

  I=ordena(I, m); // Ordenar en base al valor de cada objetivo

  $I[1]_{distancia} = I[l]_{distancia} = \infty$; // Puntos en el límite

  for $i = 2$ **to** $(l - i)$ **do** // Para todos los demás puntos en I

    $I[i]_{distancia} = I[i]_{distancia} + (I[i + 1].m - I[i - 1].m)/(f_m^{max} - f_m^{min})$

  end

end

Figura 4.3: Asignación de distancia crowding

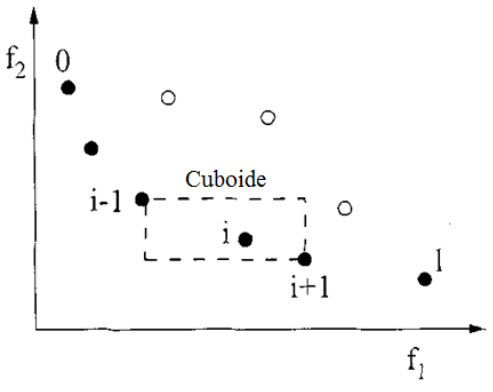

Figura 4.4: Cálculo de la distancia crowding

4.4.4. Operador de atracción

En el operador de atracción, la mejor bacteria del archivo externo (θ^{BCr}) atrae a las bacterias de la población. Como son un conjunto de soluciones no dominadas se escoge como mejor aquella que tenga el mayor valor de distancia crowding. El nuevo operador de atracción es:

$$\theta^i(j+1,G) = \theta^i(j,G) + \beta(\theta^{BC_r}(G) - \theta^i(j,G))$$

Si el archivo está vacío, el líder se toma de la población actual tomando en cuenta las reglas de factibilidad, esto con la finalidad de encontrar la región factible.

4.4.5. Reproducción

Debido a que el proceso de reproducción puede decrementar la diversidad de la población, la cual es necesaria en problemas multiobjetivo, en el ciclo de reproducción únicamente la mejor bacteria se copia y reemplaza a la segunda peor.

El pseudocódigo de MOMBFOA puede observarse en la Fig. 4.5.

A continuación se describe el pseudocódigo de MOMBFOA:

1. Se inicializan los parámetros con los que trabajará el algoritmo, los cuales son número de bacterias S_b, límite del paso quimiotáxico N_c, factor de escalamiento β, porcentaje de tamaño de paso R y el número de generaciones $GMAX$.

2. Se procede a generar una población aleatoria de bacterias $\theta^i(j,0) \forall i, i = 1, ..., S_b$ el tamaño de la población permanece constante durante toda la ejecución del algoritmo.

3. Se evalúa cada una de las bacterias $f_k(\theta^i(j,0))$, $g_m(\theta^i(j,0))$, $\forall i, i = 1, ..., S_b$, $\forall k, k = 1, ..., K, \forall m, m = 1, ..., M$, donde cada bacteria es sustituida en cada una de las funciones objetivo y restricciones del problema.

4. Se agregan las soluciones factibles no dominadas al archivo.

5. Se incrementa en 1 el contador de generaciones que estará en ejecución hasta el número máximo establecido en los parámetros. En cada generación se llevará a cabo para cada una de las bacterias el proceso quimiotáxico N_c veces, un proceso de reproducción y un proceso de eliminación-dispersión en la población de bacterias.

6. Para cada bacteria se realiza el ciclo quimiotáxico N_c veces (establecido en los parámetros inicializados), en el cual cada una de las bacterias inicia su

```
Inicializar parámetros
Archivo={}
Crear una población inicial aleatoria de bacterias θ^i(j,0) ∀i, i = 1, ..., S_b
Evaluar f_k(θ^i(j,0)), g_m(θ^i(j,0)),
  ∀i, i = 1, ..., S_b, ∀k, k = 1, ..., K, ∀m, m = 1, ..., M
for i = 1 to S_b do
    if θ^i(j,0) es factible then
        Agregar θ^i(j,0) al archivo usando dominancia de Pareto
    end
end
Computar la distancia crowding para todas las soluciones en el Archivo
for G = 1 to GMAX do
    for i = 1 to S_b do
        for j = 1 to N_c do
            Llevar a cabo el proceso quimiotáxico y el operador de atracción
            para la bacteria θ^i(j,G)
        end
    end
    for j = 1 to S_b do
        Si la bacteria θ^i(j,G) es factible, agregarla al archivo
    end
    Verificar dominancia de Pareto en archivo
    Calcular distancia crowding en archivo
    Llevar a cabo el proceso de reproducción duplicando la mejor bacteria
    de la población y eliminando la penúltima
    Llevar a cabo el proceso de eliminación- dispersión eliminando la peor
    bacteria θ^w(j,G) en la población actual y generando una aleatoriamente
end
```

Figura 4.5: MOMBFOA. Los parámetros de entrada son el número de bacterias S_b, límite del paso quimiotáxico N_c, factor de escalamiento β, porcentaje del tamaño de paso R y el número de generaciones $GMAX$. Donde K corresponde al número de funciones objetivo.

búsqueda, nadando y avanzando hacia una nueva posición si la dirección de búsqueda aleatoria genera una mejor solución; o girando para generar una nueva dirección de búsqueda si su avance no fue favorable en la búsqueda de nutrientes, es decir no se obtuvo una solución (posición de la bacteria) factible no dominada.

7. Una vez terminado el ciclo quimiotáxico el archivo externo es actualizado con soluciones factibles encontradas por las bacterias, las cuales son sometidas al concepto de dominancia de Pareto y se evalúa su distancia crowding para así obtener el archivo externo ordenado de manera descendente en referencia a la distancia crowding, que será utilizado en la próxima generación por el ciclo quimiotáxico.

8. Se realiza el proceso de reproducción, donde sólo se reproduce la mejor bacteria y se elimina a la antepenúltima bacteria de acuerdo a las reglas de factibilidad y el concepto de dominancia de Pareto.

9. Se realiza el proceso de eliminación-dispersión en el cual se elimina únicamente a la peor bacteria de la población $\theta^w(j, G)$, basados en la reglas de factibilidad y se genera una nueva bacteria aleatoriamente.

10. Se realiza otra vez el proceso desde el paso número 5 hasta cumplir con el número máximo de generaciones.

4.5. Híbrido MOMBFOA - DE

Intentando mejorar el desempeño de BFOA se han propuesto algunos cambios al algoritmo original.

Una de estas modificaciones ha sido combinar el BFOA original con el algoritmo de Evolución Diferencial[47], el cual es un algoritmo que ha sido ampliamente utilizado para la resolución de problemas de optimización global con buenos resultados. Algunos de los intentos reportados en la literatura especializada sobre este tipo de combinación son los siguientes: [5], donde cada bacteria realiza sus pasos quimiotáxicos y posteriormente se le aplica mutación diferencial, y [26] en el cual la población de bacterias realiza sus ciclos quimiotáxicos y una vez finalizado este proceso, se le aplican operadores de evolución diferencial. La aplicación de esta hibridación mejoró el desempeño del BFOA original. Cabe destacar que estos algoritmos fueron diseñados para resolver problemas de optimización de un solo objetivo.

Persiguiendo la misma meta de mejorar el desempeño del mecanismo de búsqueda de MOMBFOA en espacios con un número elevado de dimensiones, como lo es el problema de la cadena de suministro que se resuelve en este trabajo, se optó por

```
1 d_rand = rand[1, D]
2 for d = 1 to D do
3 |   if rand[0, 1] < CR or d = d_rand then
4 |   |   θ^i_R(j + 1, G)[d] = θ^r0(j, G)[d] + F(θ^r1(j, G)[d] − θ^r2(j, G)[d])
5 |   else
6 |   |   θ^i_R(j + 1, G)[d] = θ^i(j, G)[d]
7 |   end
8 end
```

Figura 4.6: Cruza de Evolución Diferencial. D es la dimensionalidad del problema, CR la taza de cruza, $\theta^i_R(j+1,G)$ es el vector trial, $\theta^{r0}(j,G), \theta^{r1}(j,G)$ y $\theta^{r2}(j,G)$ son vectores tomados aleatoriamente del archivo (si el tamaño del archivo es menor a 3 se toman los necesarios de la población).

utilizar dentro del algoritmo MOMBFOA mecanismos de búsqueda de Evolución Diferencial.

Los principales mecanismos de búsqueda en Evolución Diferencial son la mutación y la cruza, los cuáles aplicados a las bacterias son:

Mutación:

$$\theta^i_R(j+1,G) = \theta^{r0}(j,G) + F(\theta^{r1}(j,G) - \theta^{r2}(j,G))$$

La cual se aplica durante la cruza, ver Fig. 4.6.

La modificación que se realizó sobre el MOMBFOA original consistió en cambiar el proceso de paso quimiotáxico que llevan a cabo las bacterias.

El paso quimiotáxico del MOMBFOA modificado que realiza cada bacteria durante su ciclo quimiotáxico consiste de los siguientes pasos:

1. Al principio de cada paso quimiotáxico, se le aplica a la bacteria $\theta^i(j,G)$ los mecanismos de mutación y cruza de evolución diferencial, de esta manera se obtiene una bacteria trial $\theta^i_R(j,G)$.

2. Una vez que tenemos el vector trial, se le aplica el operador de giro-nado original, de esta manera el vector trial $\theta^i_R(j,G)$ obtiene nuevos valores.

3. El siguiente paso consiste en evaluar el vector trial en la función objetivo

4. Si el vector trial tiene mejor valor en la función objetivo, sustituye a la bacteria original, en caso contrario la original se conserva.

El pseudocódigo de MOMBFOA-DE se puede ver en la Fig. 4.7.

```
Inicializar parámetros
Archivo={}
Crear una población inicial aleatoria de bacterias θ^i(j,0) ∀i, i = 1, ..., S_b
Evaluar f_k(θ^i(j,0)), g_m(θ^i(j,0)),
  ∀i, i = 1, ..., S_b, ∀k, k = 1, ..., K, ∀m, m = 1, ..., M
for i = 1 to S_b do
    if θ^i(j,0) es factible then
        Agregar θ^i(j,0) al archivo usando dominancia de Pareto
    end
end
Computar la distancia crowding para todas las soluciones en el Archivo
for G = 1 to GMAX do
    for i = 1 to S_b do
        for j = 1 to N_c do
            Llevar a cabo mutación y cruza de evolución diferencial
            para obtener θ^i_R(j,G)
            Llevar a cabo el proceso quimiotáxico y el operador de
            atracción para la bacteria θ^i_R(j,G)
            θ^i(j+1,G) = mejor(θ^i(j,G), θ^i_R(j,G))
        end
    end
    for j = 1 to S_b do
        Si la bacteria θ^i(j,G) es factible, agregarla al archivo
    end
    Verificar dominancia de Pareto en archivo
    Calcular distancia crowding en archivo
    Llevar a cabo el proceso de reproducción duplicando la mejor bacteria
    de la población y eliminando la penúltima
    Llevar a cabo el proceso de eliminación- dispersión eliminando la peor
    bacteria θ^w(j,G) en la población actual y generando una aleatoriamente
end
```

Figura 4.7: MOMBFOA-DE. Los parámetros de entrada son el número de bacterias S_b, límite del paso quimiotáxico N_c, factor de escalamiento β, porcentaje del tamaño de paso R, el número de generaciones $GMAX$, el factor de cruza de ED CR y el factor de escalamiento de ED F. Donde K corresponde al número de funciones objetivo.

Capítulo 5

Experimentación

El modelo formal de la instancia del problema de la cadena de suministros, descrito en la sección 3.3, fue implementado y se le aplicaron tres algoritmos de optimización multiobjetivo para resolverlo. Los algoritmos utilizados fueron MOMBFOA-DE, del cual se habló en la sección anterior, NSGA-II y SPEA2 los cuales serán descritos a continuación.

5.1. NSGA-II

NSGA-II fue propuesto en [17] en el 2000, como una mejora a su antecesor el NSGA, para corregir algunas desventajas que presentaba:

- *Alta complejidad computacional del ordenamiento no dominado:* el ordenamiento no dominado que lleva a cabo el NSGA original tiene una complejidad de $O(MN^3)$ (donde M es el número de objetivos y N es el tamaño de la población). Esto hace a NSGA muy costoso con tamaños de población grandes.
- *Falta de elitismo:* el elitismo incrementa significativamente el desempeño de los algoritmos genéticos, también previene de la pérdida de buenas soluciones una vez que se encuentran.
- *Necesidad de especificar el parámetro sharing* σ_{share}: mecanismos tradicionales para mantener la diversidad de la población y tener amplia variedad de soluciones han dependido del concepto de compartir. El problema principal de ésto es que se necesita especificar el parámetro σ_{share}.

5.1.1. Ordenamiento rápido no dominado

La manera en que NSGA-II lleva a cabo el ordenamiento es la siguiente. Primero, cada solución tiene dos entidades: 1) Un contador de dominación n_p, que

es el número de soluciones las cuales dominan a la solución p, y 2) S_p, que es un conjunto de soluciones que la solución p domina.

Todas las soluciones en el primer frente no dominado tendrán n_p en 0. Ahora, por cada solución p con $n_p = 0$, visitamos cada miembro q de su conjunto s_p y se reduce su contador de dominación en una unidad, si en algún miembro q s_p se vuelve 0, se coloca en una lista separada Q. Esos miembros pertenecen al segundo frente no dominado. Se continúa el proceso con cada miembro de Q y el tercer frente es identificado y así sucesivamente hasta encontrar todos los frentes.

El pseudocódigo de este proceso de ordenamiento se muestra en la Fig. 5.1.

5.1.2. Preservación de la diversidad

Durante el procesamiento del algoritmo, es deseable se tengan las soluciones dispersas tanto en el espacio de búsqueda como en el frente de Pareto que se va generando. En NSGA-II se utiliza la distancia de crowding para estimar la densidad de soluciones alrededor de una solución en particular. Esta distancia es un estimado del perímetro del cuboide formado usando los vecinos más cercanos como vértices, todo en el espacio de los objetivos. En la Fig. 4.4, la distancia crowding de la solución i en su frente (círculos en negro) es la longitud promedio de los lados del cuboide (formado por líneas punteadas). El pseudocódigo de este proceso se muestra en la Fig. 4.3 en donde se computan todas las soluciones en un conjunto no dominado I. $I[i].m$ se refiere al valor de la m-ésima función objetivo del i-ésimo individuo en el conjunto I y los parámetros f_m^{max} y f_m^{min} son el máximo y mínimos valores de la m-ésima función objetivo.

Una vez que se han asignado las distancias de crowding se puede aplicar el operador de Comparación-crowded ($\prec_n$) el cual guía el proceso de selección en varias etapas del algoritmo. Este operador asume que cada individuo tiene dos atributos: 1) rango de dominancia (i_{rank}) y 2) distancia crowding ($i_{distance}$), entonces se puede definir un orden parcial $\prec_n$ como

$$\begin{gathered} i \prec_n j \text{ si } (i_{rank} < j_{rank}) \\ \text{o } ((i_{rank} = j_{rank}) \text{ y } (i_{distance} > j_{distance})) \end{gathered}$$

5.1.3. Ciclo principal

El ciclo principal del NSGA-II es el siguiente. Primero, se crea una población R_t combinando los padres y los hijos $P_t \cup Q_t$. Esta población tiene un tamaño de $2N$. Entonces, la población se ordena de acuerdo a la no dominancia. Una vez hecho el ordenamiento se tendrá un conjunto de frentes no dominados F, donde el mejor frente será F_1. La nueva población de tamaño N se formará con los individuos de los mejores frentes, es decir, se copian los individuos de F_1, si no se ha llenado la población se copian los individuos de F_2, así sucesivamente hasta que ya no se puedan acomodar más individuos de los frentes. Los individuos

```
   input: Población P
 1 foreach p ∈ P do
 2 |   S_p = {}
 3 |   n_p = 0
 4 |   foreach q ∈ P do
 5 |   |   if p ≼ q then // Si p domina q
 6 |   |   |   S_p = S_p ∪ {q} ;  // Agrega q a soluciones dominadas por p
 7 |   |   else if q ≼ p then
 8 |   |   |   n_p = n_p + 1 ;   // Incrementa contador de dominación de p
 9 |   |   end
10 |   end
11 |   if n_p = 0 then // p pertenece al primer frente
12 |   |   P_rank = 1
13 |   |   F_1 = F_1 ∪ {p}
14 |   end
15 end
16 i = 1 ;                              // Inicializa el contador de frente
17 while F_i ≠ {} do
18 |   Q = {} ; // Se usa para guardar miembros del siguiente frente
19 |   foreach p ∈ F_i do
20 |   |   foreach q ∈ S_p do
21 |   |   |   n_q = n_q − 1
22 |   |   |   if n_q = 0 then // q pertenece al siguiente frente
23 |   |   |   |   q_rank = i + 1
24 |   |   |   |   Q = Q ∪ {q}
25 |   |   |   end
26 |   |   end
27 |   end
28 |   i = i + 1
29 |   F_i = Q
30 end
```

Figura 5.1: Ordenamiento rápido no dominado

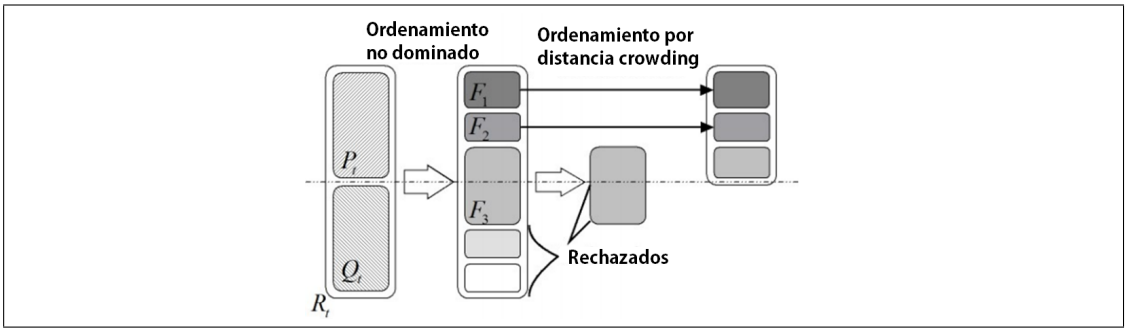

Figura 5.2: Procedimiento del NSGA-II

1 $R_t = P_t \cup Q_t$; // Combina padres y descendencia
2 F = ordenamientoRapidoNoDominado(R_t); // F =Frentes en R_t
3 $P_{t+1} = \{\}$
4 $i = 1$
5 **while** $P_{t+1} + F_i \leq N$ **do** // Hasta que la población de padres se llene
6 | asignarCrowdingDistance(F_i)
7 | $P_{t+1} = P_{t+1} \cup F_i$
8 | $i = i + 1$
9 **end**
10 ordena($F_i, \prec_n$); // Ordena en descenso usando $\prec_n$
11 $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$; // Solo elementos necesarios para llenar la población
12 Q_{t+1} =crearDescendencia(P_{t+1}); // Selección, cruza y mutación
13 $t = t + 1$

Figura 5.3: Ciclo principal de NSGA-II

faltantes se toman de la siguiente manera: se ordenan mediante el operador $\prec_n$ los individuos del último frente visitado (el que ya no se pudo acomodar) y se toman los primeros individuos necesarios para llenar la población.

Este proceso se puede observar gráficamente en la Fig. 5.2 la cual muestra que en un principio se le aplica ordenamiento rápido no dominado a la población de padres y descendencia, de esta manera se obtienen los distintos frentes de Pareto. Posteriormente, se les asigna distancia crowding a los individuos para finalmente obtener la población que pasa a la siguiente generación.

El pseudocódigo de NSGA-II se observa en la Fig. 5.3.

5.2. SPEA2

El algoritmo SPEA2 fue propuesto en [53] en el 2001, como mejora al Strength Pareto Evolutionary Algorithm (SPEA). Las principales diferencias con su antecesor son las siguientes:

- Se usa un esquema de asignación de fitness mejorado, el cual toma en cuenta, para cada individuo, cuántos individuos domina y por cuántos es dominado.
- Se incorpora una técnica de estimación de densidad del vecino más cercano, la cual permite una guía más precisa del proceso de búsqueda.
- Un método de truncado de archivo que garantiza la preservación de las soluciones en la frontera.

SPEA2 utiliza un tamaño de población N fijo y un archivo (conjunto externo) cuyo tamaño $\bar{N}$ también se mantiene fijo. El algoritmo se ejecuta hasta alcanzar un máximo número de generaciones T. El pseudocódigo del algoritmo se puede observar en la Fig. 5.4.

El algoritmo de SPEA2 se desarrolla a lo largo de las siguientes etapas:

1. Inicialización: en esta etapa se genera una población inicial P_0, se crea el archivo vacío (conjunto externo) $\bar{P}_0 =$ y se establece el contador de tiempos $t = 0$
2. Asignación de fitness: en esta etapa se calculan los valores de fitness de los individuos en P_t y $\bar{P}_t$
3. Selección ambiental: durante esta etapa se copian todos los individuos no dominados en P_t y $\bar{P}_t$ a $\bar{P}_{t+1}$. Si el tamaño de $\bar{P}_{t+1}$ excede $\bar{N}$ entonces se reduce $\bar{P}_{t+1}$ por medio del operador de truncado, en otro caso si el tamaño de $\bar{P}_{t+1}$ es menor que $\bar{N}$ entonces se llena $\bar{P}_{t+1}$ con individuos dominados en P_t y $\bar{P}_t$.
4. Terminación: si $t \geq T$ o cualquier otro criterio de parada se satisface entonces establecer A igual al conjunto de vectores de decisión representados por los individuos no dominados en $\bar{P}_{t+1}$. Se detiene la ejecución del algoritmo.
5. Selección de padres: se lleva a cabo torneo binario de selección con reemplazo sobre $\bar{P}_{t+1}$ para la obtención de los padres.
6. Variación: se aplican operadores de recombinación y mutación a los padres y se establece P_{t+1} con la población resultante. Se incrementa t y se vuelve a la etapa de Asignación de fitness.

input : N, $\bar{N}$, T ; `// Tamaño de población, tamaño de archivo y máximo número de generaciones`
output: A ; `// conjunto de soluciones no dominadas`
1 Crear población aleatoria P_0, $\bar{P}_0 = \{\}$, $t = 0$
2 Evaluar soluciones en P_t y $\bar{P}_t$
3 Copiar individuos no dominados de P_t y $\bar{P}_t$ a $\bar{P}_{t+1}$.
4 **if** $t \geq T$ **then**
5 | $A = \bar{P}_{t+1}$
6 | Terminar
7 **end**
8 Obtener padres mediante torneo binario de selección con reemplazo sobre $\bar{P}_{t+1}$
9 Establecer P_{t+1} aplicándole a los padres operadores de recombinación y mutación. Incrementar t. Ir a 2

Figura 5.4: Pseudocódigo de SPEA2

5.2.1. Asignación de fitness

SPEA2 utiliza una asignación fitness de grano fino, para cada solución se toma en cuenta el número de soluciones que domina y por cuántas es dominada. Para ello asigna a cada individuo i en el archivo $\bar{P}_t$ y la población P_t un valor de fuerza $S(i)$, representando el número de soluciones que domina:

$$S(i) = |\{j | j \in P_t + \bar{P}_t \wedge i \preceq j\}|$$

Tomando como base los valores de S un fitness primario $R(i)$ de un individuo se calcula:

$$R(i) = \sum_{j \in P_t + \bar{P}_t, j \preceq i} S(j)$$

Este fitness primario se determina sumando las fuerzas S de sus dominadores en el archivo y la población. Un $R(i) = 0$ significa que el individuo es una solución no dominada mientras que un valor alto representa que es dominado por muchos individuos.

Aunque el fitness primario facilita un mecanismo de ordenamiento basado en la dominancia de Pareto, puede fallar cuando muchos individuos no se dominan entre ellos. Por lo tanto información adicional sobre la densidad se incorpora para discriminar entra individuos que tengan idénticos valores de fitness primario.

Para calcular la densidad de cada individuo i, primero, se miden las distancias hacia cada individuo j del archivo y la población y se guardan en una lista ordenada. Una vez que se tiene esa lista ordenada de menor a mayor, el k ésimo

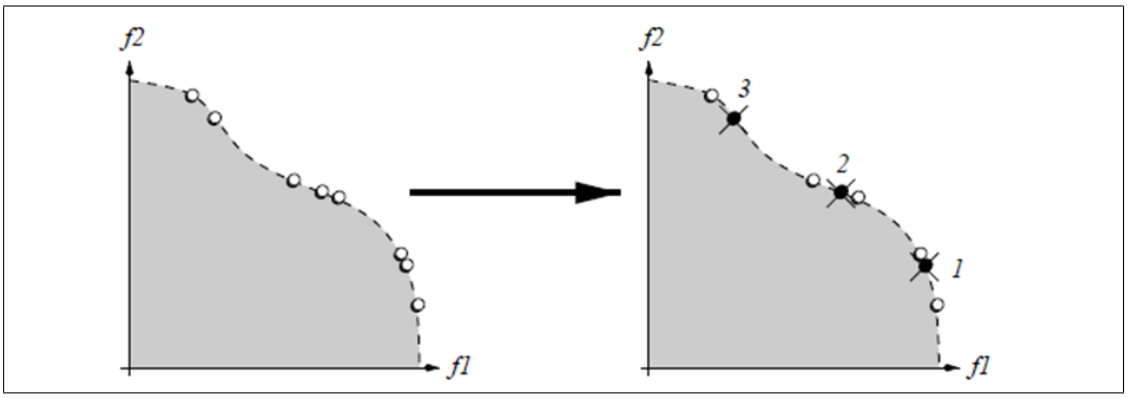

Figura 5.5: Método de truncado en SPEA2, asumiendo $\bar{N} = 5$. En la izquierda el conjunto de soluciones no dominadas, a la derecha las soluciones que son eliminadas y el orden en que se hizo.

elemento proporciona la distancia buscada σ_i^k, usando $k = \sqrt{N + \bar{N}}$. Posteriormente se calcula la densidad:

$$D(i) = \frac{1}{\sigma_i^k + 2}$$

En el denominador se agrega 2 para asegurarse que su valor es mayor a cero y menor a 1. Finalmente el fitness $F(i)$ de un individuo se calcula:

$$F(i) = R(i) + D(i)$$

5.2.2. Selección ambiental

En la etapa de selección ambiental, el primer paso es copiar, al archivo de la siguiente generación, todos los individuos que tengan un fitness menor a uno, del archivo y de la población:

$$\bar{P}_{t+1} = \{i | i \in P_t + \bar{P}_t \wedge F(i) < 1\}$$

Si el frente no dominado es exactamente del tamaño del archivo ($|\bar{P}_{t+1}| = \bar{N}$) el paso de selección ambiental está completo. Si el archivo es muy pequeño ($|\bar{P}_{t+1}| < \bar{N}$) los mejores $\bar{N} - |\bar{P}_{t+1}|$ individuos dominados en el archivo y población previos se copian al nuevo archivo. Si el archivo es muy grande ($|\bar{P}_{t+1}| > \bar{N}$) se lleva a cabo un procedimiento de truncado el cual iterativamente va eliminando individuos del archivo hasta que su tamaño sea igual a $\bar{N}$. El individuo que se elige para ser eliminado es aquel que presenta la mínima distancia a otro individuo. La manera de llevar a cabo esta eliminación se ilustra en la Fig. 5.5.

5.3. Métricas de desempeño

Ya que los algoritmos evolutivos de optimización multiobjetivo arrojan múltiples soluciones en lugar de solo una, como lo es para el caso mono-objetivo, se hacen necesarias métricas de desempeño propias para este tipo de algoritmos. En este trabajo se utilizaron dos métricas para la comparación.

5.3.1. Two-Set Coverage

En esta métrica binaria se compara la cobertura de dos conjuntos de soluciones de los vectores A' y A''. La métrica de cobertura $C(A', A'')$ calcula el número de soluciones en A'' que son dominadas por soluciones de A' [25]:

$$C(A', A'') = \frac{|a'' \in A''| \exists a' \in A' : a' \prec a''|}{|A''|} \tag{5.1}$$

Un valor de $C(A', A'') = 1$ indica que todas las soluciones de A'' son dominadas por las de A'. Si $C(A', A'') = 0$ todas las soluciones de A' son dominadas por las de A''.

5.3.2. Hypervolume

La métrica Hypervolume mide el volumen (área para el caso de 2 dimensiones) del espacio objetivo que es cubierto por un conjunto de soluciones Q. Por cada solución $i \in Q$ se construye un hipercubo v_i con un punto de referencia W, que puede ser el valor máximo de cada función objetivo. La unión de los hipercubos es el Hypervolume Hv y se calcula mediante [25]:

$$Hv = (\bigcup_{i=1}^{|Q|} v_i) \tag{5.2}$$

En el caso de 2 dimensiones, el Hv se calcula mediante la unión de todas las áreas rectangulares dado un punto de referencia, como se observa en la Fig. 5.6.

Si W se coloca en el origen, un valor menor de Hv indica un mejor desempeño del algoritmo. Por el contrario, si W se coloca sobre los valores máximos de cada función objetivo, un mayor valor de Hv indicará mejor desempeño. En este trabajo de tesis se utilizó el segundo caso.

5.4. Diseño experimental

Se procedió a resolver la instancia del problema de la cadena de suministro, aplicando los algoritmos MOMBFOA-DE, NSGA-II[18] y SPEA2[19].

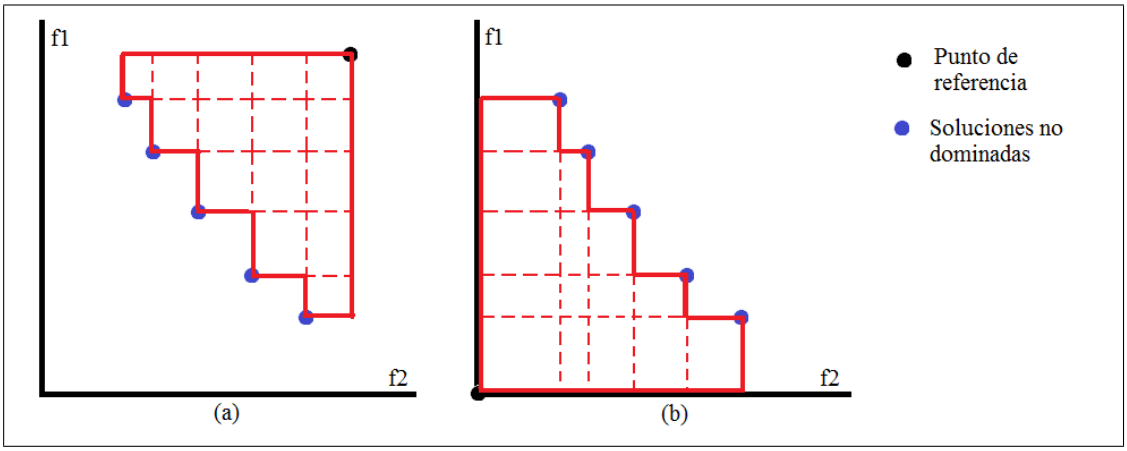

Figura 5.6: El Hypervolume es la medida del área delimitada por la línea continua, en el caso (a) dado un punto de referencia y (b) punto de referencia centrado en el origen.

Para NSGA-II y SPEA2 se utilizó una codificación binaria de las soluciones. En el caso de MOMBFOA-DE, ya que está diseñado para trabajar en espacios continuos, la codificación utilizada internamente por el algoritmo fue real y se redondeaban a números enteros en el momento de evaluar el valor de las variables en la función objetivo y las restricciones del problema.

Se llevó a cabo un experimento, en el cual se hicieron 30 corridas independientes de cada algoritmo sobre el problema de la cadena de suministro abordado en este trabajo.

En cada corrida de cada algoritmo se realizaron 50000 evaluaciones de soluciones.

Los parámetros utilizados por los algoritmos fueron calibrados con la herramienta irace [33] y se muestran en la Tabla 5.1.

Una vez obtenidos los resultados de cada algoritmo, 30 frentes de Pareto por cada uno, se procedió a realizar su comparación, aplicando dos métricas de desempeño (Two-Set Coverage e Hypervolume) y se utilizó la prueba no-paramétrica de Wilcoxon con 95 % de confianza para validar si existían diferencias significativas entre los resultados arrojados en las muestras de ejecuciones realizadas.

5.5. Comparación de resultados

Una vez obtenidos los resultados se procedió a compararlos mediante métricas de desempeño de algoritmos multiobjetivo: Two-Set Coverage e Hypervolume [37]. Esta comparación se dividió en dos partes:

En la primera se obtuvieron los valores de Hypervolume de los 90 resultados y el valor de Two-Set Coverage de cada uno de los algoritmos en cada corrida (MOMBFOA-DE - NSGA-II, NSGA-II - MOMBFOA-DE, MOMBFOA-DE -

Tabla 5.1: Valores de los parámetros de los algoritmos utilizados

MOMBFOA-DE	
Generaciones	150
Población	100
Ciclos quimiotáxicos	2
Porcentaje de tamaño de paso	0.21
Factor de escalamiento	0.001
Constante de cruza de ED (CR)	0.63
Factor de escalamiento de ED (F)	0.45
NSGA-II	
Población	100
Generaciones	500
Probabilidad de cruza	0.85
Probabilidad de mutación	0.18
SPEA2	
Población	100
Tamaño de archivo	100
Generaciones	500
Probabilidad de cruza	0.92
Probabilidad de mutación	0.07

Tabla 5.2: Estadísticas sobre los valores de Hypervolume obtenidos de las 30 corridas de cada algoritmo. Un mejor resultado se indica en **negritas.**

	MOMBFOA-DE	NSGA-II	SPEA2
Media	**423918.096**	411585.801	414379.866
Mediana	**423633.68**	414915.433	415574.347
Desv. Est.	**5023.90038**	12219.2446	9125.72076
Mejor	**433261.245**	428486.917	427103.675
Peor	**410853.599**	374376.434	388436.761

Tabla 5.3: Prueba de Wilcoxon a valores de Hypervolume.

MOMBFOA-DE - NSGA-II	1.49E-06
MOMBFOA-DE - SPEA2	8.29E-06
NSGA-II - SPEA2	0.5592

Tabla 5.4: Estadísticas sobre los valores de Two-Set Coverage obtenidos de las 30 corridas de cada algoritmo.

	MOMBFOA-DE NSGA-II	NSGA-II MOMBFOA-DE	MOMBFOA-DE SPEA2	SPEA2 MOMBFOA-DE	NSGA-II SPEA2	SPEA2 NSGA-II
Media	0.5260264	0.451971962	0.37367058	0.48420854	0.5837112	0.6157922
Mediana	0.5321232	0.358952703	0.27243867	0.44949494	0.6614945	0.8203957
Desv. Est.	0.2725556	0.44072505	0.3149485	0.38310675	0.4105041	0.4288629
Mejor	1	1	1	1	1	1
Peor	0	0	0	0	0	0
Wilcoxon	0.4454		0.3575		0.8203	

SPEA2, SPEA2 - MOMBFOA-DE, NSGA-II - SPEA2, SPEA2 - NSGA-II). Con base en estos valores de las métricas se obtuvieron el mejor valor, el peor, la media, mediana y desviación estándar. También se aplicó la prueba de Wilcoxon para observar si existían diferencias significativas entre los resultados de los algoritmos. En la Tabla 5.2 se muestran las estadísticas sobre los valores de Hypervolume obtenidos en las 30 corridas, mientras que en la Tabla 5.3 están los resultados de la prueba de Wilcoxon aplicada a los valores de Hypervolume de las 30 corridas, un valor $<$ a 0.05 indica que existen diferencias significativas entre las muestras. En la Tabla 5.4 se observan los resultados de las estadísticas sobre los valores de Two-Set Coverage en las 30 corridas.

La segunda parte consistió en obtener el frente de Pareto acumulado de las 30 ejecuciones independientes. Una vez obtenidos los 3 frentes acumulados, uno por cada algoritmo, se compararon mediante Hypervolume y Two-Set Coverage. Los frentes acumulados enfrentados se pueden observar en la Fig. 5.7. Los resultados de Hypervolume se encuentran en la Tabla 5.5 y los resultados de Two-Set Coverage se observan en la Tabla 5.6.

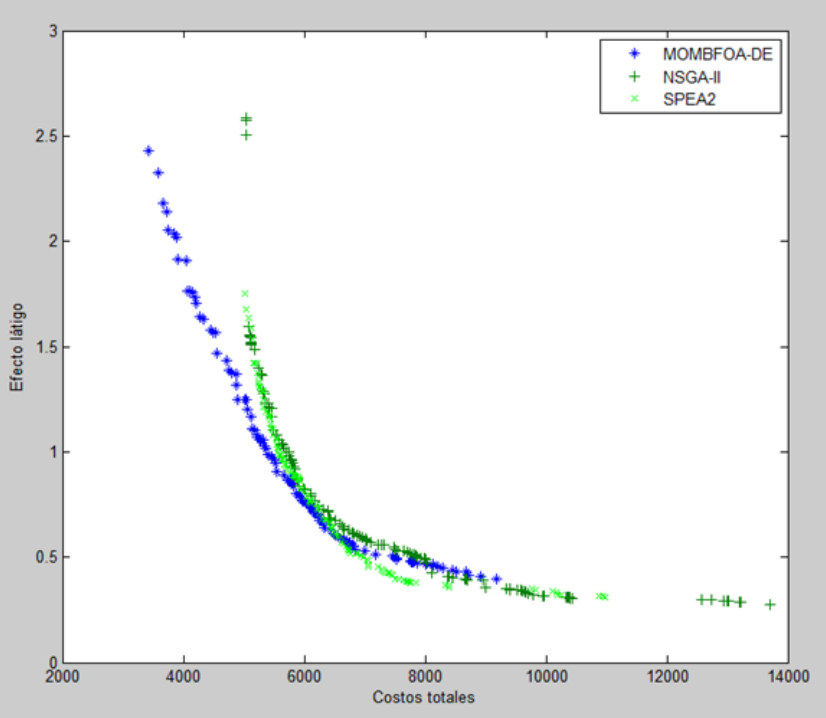

Figura 5.7: Frentes acumulados de 30 corridas. Un * representa los puntos del frente de Pareto de MOMBFOA-DE, + los puntos del frente de Pareto de NSGA-II y x señala los puntos del frente de Pareto de SPEA2.

Tabla 5.5: Valores de Hypervolume de los frentes acumulados. Un mejor resultado se indica en **negritas**

MOMBFOA-DE	**24512.0607**
NSGA-II	22528.3392
SPEA2	22791.0256

Tabla 5.6: Valores de Two-Set Coverage de los frentes acumulados. Un mejor resultado se indica en **negritas**

	MOMBFOA-DE	NSGA-II	SPEA2
MOMBFOA-DE	–	**0.66242038**	**0.59036145**
NSGA-II	0.09345794	–	0.15060241
SPEA2	0.26168224	**0.65605096**	–

Tabla 5.7: Valores de las funciones objetivo de las tres soluciones extraídas de los frentes acumulados.

Algoritmo	Costos totales	Efecto látigo
MOMBFOA-DE	6029.5	0.754995306975249
NSGA-II	6106	0.803291418
SPEA2	6060.5	0.7875281260045002

5.6. Discusión de resultados

Dados los resultados de las métricas de desempeño y las estadísticas, se observa que MOMBFOA-DE tuvo un desempeño ligeramente superior a los otros dos algoritmos, pues, en el caso de Hypervolume, presentó un valor mayor en las estadísticas de las 30 corridas y también de los frentes acumulados y la prueba de Wilcoxon arrojó que tenía diferencias significativas con los otros dos algoritmos, en cambio, la prueba arrojó también que no existían diferencias entre los valores de Hypervolume de NSGA-II y SPEA2.

En el caso de Two-Set Coverage, en las estadísticas de las 30 corridas independientes, MOMBFOA-DE tuvo mejor desempeño que NSGA-II, y SPEA2 arrojó mejores resultados que los otros dos algoritmos. Sin embargo, al comparar los resultados con la prueba de Wilcoxon, ésta dice que no existen diferencias significativas entre los algoritmos.

En los frentes acumulados, Two-Set Coverage indica un mejor desempeño de MOMBFOA-DE sobre los otros dos algoritmos, y mejor desempeño de SPEA2 comparado con NSGA-II.

De los frentes acumulados se obtuvieron tres soluciones que se encontraban en la parte central (ver la Fig. 5.8). Los valores de cada solución en el espacio de funciones objetivo se encuentra en la Tabla 5.7.

A pesar de que la solución de MOMBFOA-DE domina a las otras dos soluciones no existe una diferencia importante entre ellas, como se observa en la Fig. 5.8.

Con respecto del problema, todas las restricciones se cumplieron y se logró una reducción de costos totales y efecto látigo, este último mostró niveles muy bajos, casi alcanzó a desaparecer.

Las soluciones arrojan que los niveles de inventario deben mantenerse cerca del mínimo, pero con suficiente margen para no descuidar un aumento en las futuras órdenes, esto dependerá del tipo de demanda de la cadena. Con ésto, los costos por mantenimiento se mantienen bajos, mientras la demanda de los mayoristas se cumple y el efecto látigo se mantiene en niveles mínimos.

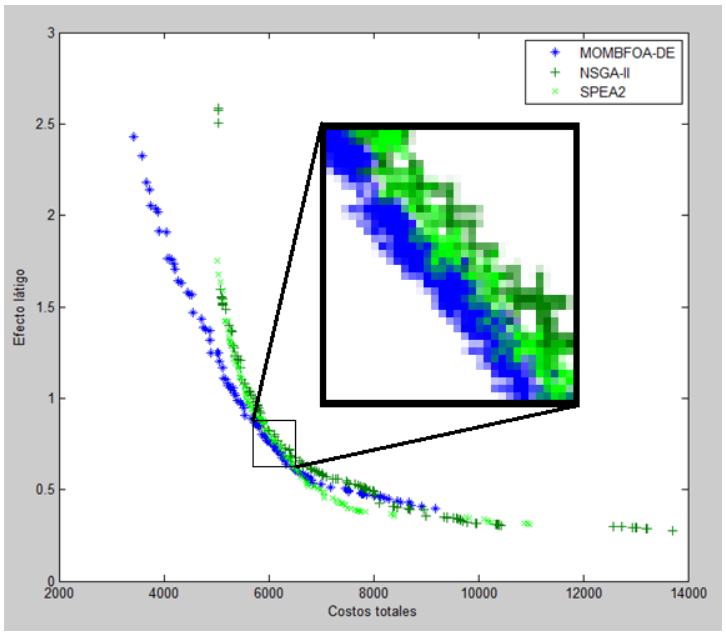

Figura 5.8: Sección de los frentes acumulados de la cual se tomaron las tres soluciones representativas, una por cada algoritmo.

Capítulo 6

Conclusiones y trabajos futuros

Este trabajo presentó una comparativa de algoritmos evolutivos multiobjetivo en una instancia del problema de la cadena de suministro, donde se requería minimizar los costos totales de operación al mismo tiempo que el efecto látigo y estaba sujeta a una serie de restricciones de operación.

Los algoritmos utilizados fueron: el algoritmo basado en el forrajeo de bacterias multiobjetivo combinado con el operador de mutación de Evolución Diferencial (MOMBFOA-DE) y dos algoritmos multiobjetivo del estado del arte (NSGA-II y SPEA2).

Los resultados arrojados por los algoritmos se compararon mediante métricas de desempeño para algoritmos evolutivos multiobjetivo: Two-Set Coverage e Hypervolume. Además se aplicaron pruebas estadísticas para la validación de los resultados.

Los resultados de los algoritmos mostraron los niveles de inventario y el número de unidades de producto transportadas del distribuidor a los mayoristas que son necesarios para que los costos totales de operación y el efecto látigo se reduzcan. En los tres casos los frentes de Pareto acumulados muestran un buen compromiso entre costos totales y efecto látigo. En la parte central de estos frentes acumulados, los algoritmos tuvieron un comportamiento similar, aunque en los extremos se observan diferencias, las cuales son soluciones que dan prioridad a uno de los objetivos, ya sea costos totales o efecto látigo y dependerá del diseñador experto tomarlas en cuenta, de acuerdo a sus intereses.

Dados los resultados de las métricas de desempeño y las estadísticas se observa que MOMBFOA-DE tuvo un desempeño ligeramente superior a los otros dos algoritmos, pues, en el caso de Hypervolume, presentó un mejor valor en las estadísticas de las 30 corridas y también de los frentes acumulados y la prueba de Wilcoxon arrojó que tenía diferencias significativas con los otros dos algoritmos, en cambio, la prueba arrojó también que no existían diferencias entre los valores de Hypervolume de NSGA-II y SPEA2.

En el caso de Two-Set Coverage, en las estadísticas de las 30 corridas in-

dependientes, MOMBFOA-DE tuvo mejor desempeño que NSGA-II, y SPEA2 arrojó mejores resultados que los otros dos algoritmos en las muestras de ejecuciones realizadas. Sin embargo, al comparar los resultados con la prueba de Wilcoxon, indicó que no existen diferencias significativas entre los algoritmos. En los frentes acumulados, Two-Set Coverage indica un mejor desempeño de MOMBFOA-DE sobre los otros dos algoritmos, y mejor desempeño de SPEA2 comparado con NSGA-II.

MOMBFOA-DE obtiene resultados competitivos e incluso mejores (en algunas métricas de desempeño) comparado con algoritmos evolutivos del estado del arte, en el problema abordado.

Como trabajos futuros se espera el utilizar MOMBFOA-DE en otras instancias de cadenas de suministro y en otros problemas de optimización multiobjetivo en general.

Para observar y medir las diferencias de desempeño entre el mecanismo de búsqueda de MOMBFOA-DE y Evolución Diferencial en problemas multiobjetivo, se requiere de una comparación entre MOMBFOA-DE y algún algoritmo de optimización multiobjetivo que utilice únicamente Evolución Diferencial como mecanismo de búsqueda, por ejemplo GDE3 [28].

Se espera también comparar MOMBFOA-DE con otros algoritmos de optimización multiobjetivo del estado del arte, por ejemplo MOEA/D [52], el cual no utiliza dominancia de Pareto en su funcionamiento.

Bibliografía

[1] R. H. Ballou. *Logística Administración de la Cadena de Suministros*. Prentice Hall, 2004.

[2] S. Bandyopadhyay and R. Bhattacharya. Appliying modified nsga-ii for bi-objective supply chain problem. *Journal of Intelligent Manufacturing*, pages 1–10, 2011.

[3] S. Bandyopadhyay and R. Bhattacharya. Applying modified nsga-ii for bi-objective supply chain problem. *Intelligent Manufacturing*, 24, 2011.

[4] B. M. Beamon. Measuring supply chain performance. *Int. J. Oper. Prod. Manage*, 19(3):275–292, 1999.

[5] A. Biswas, S. Dasgupta, and A. Abraham. A synergy of differential evolution and bacterial foraging optimization for faster global search. *International Journal on Neural ans Mass-Parallel Computing and Information Systems*, pages 607–626, 2007.

[6] J. R. Burns and B. Janamanchi. Strategies for reducing inventory costs and mitigating the bullwhip effect in supply chains: a simulation study. Southwest Division of Decision Sciences Institute conference, 2006.

[7] K. K. Castillo-Villar and J. F. Herbert-Acero. Approach for the capacitated supply chain network design problem including imperfect quality an rework. *IEEE Computational Intelligence*, 9(4):31–45, 2014.

[8] Z. Che and C. J. Chiang. A modified pareto genetic algorithm for multi-objective build-to-order supply chain planning with product assembly. *Advances in Engineering Software*, 41, 2010.

[9] C. A. Coello Coello. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academics, 2002.

[10] A. Cortés Godínez. Comparación de dos algoritmos metaheurísticos en la solución multi-objetivo de un modelo particular de la cadena de suministros. Master's thesis, Instituto Tecnológico de León, 2010.

[11] A. Cortés Godínez. Comparación de dos algoritmos metaheurísticos en la solución multi-objetivo de un modelo particular de la cadena de suministros. Master's thesis, Instituto Tecnológico de León, León, Guanajuato, 2010.

[12] A. Cortés Godínez and L. E. Mancilla Espinoza. La inteligencia artificial aplicada a la cadena de suministros. Congreso de Sistemas de Innovación para la Competitividad (SINNCO), 2010.

[13] S. Dalal and V. Athavale. Challenging bullwhip effect of supply chain through case-based multi-agent system: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2, 2012.

[14] K. Deb. *Optimization for Engineering Design Algorithms and Examples*. Prentice-Hall of India, Nueva Delhi, 1995.

[15] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

[16] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[17] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2000.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm:nsga-ii. pages 182–197. IEEE Transaction on evolutionary Computation, 2002.

[19] Z. Eckart, M. Laumanns, and L. Thiele. Spea2: Improving the strenght pareto evolutionary algorithm. Technical report, Swiss Federal Institute of Technology, 2001.

[20] A. P. Engelbrecht. *Computational Intelligence An Introduction*. Wiley, second edition, 2007.

[21] G. Ertek and E. Erylmaz. The bullwhip effect in supply chain, reflections after a decade. CELS, 2008.

[22] J. Forrester. Industrial dynamics. *MIT press Cambridge, MA*, 2, 1961.

[23] J. C. Fransoo and M. Wouters. Measuring the bullwhip effect in the supply chain. *Supply chain Management*, 5, 2000.

[24] M. Gutiérrez Contreras and L. E. Mancilla Espinoza. Multi-agentes distribuidos aplicados a la cadena de suministro. Technical report, Instituto Tecnológico de León, 2012.

[25] B. Hernández Ocaña. Diseño mecatrónico usando optimización basada en bacterias. Master's thesis, Laboratorio Nacional de Informática Avanzada, Xalapa,Ver., 2011.

[26] Y. Jarraya, S. Bouaziz, A. Alimi, and A. Abraham. A hybrid computational chemotaxis in bacterial foraging optimization algorithm for global numerical optimization. In *Cybernetics (CYBCONF), 2013 IEEE International Conference on*, pages 213–218, June 2013.

[27] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, UK, 2001.

[28] S. Kukkonen and J. Lampinen. Gde3: the third evolution step of generalized differential evolution. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, volume 1, pages 443–450 Vol.1, Sept 2005.

[29] H. L. Lee, V. Padmanabhan, and S. Whang. The bullwhip effect in supply chain. *Sloan management review*, pages 93–102, 1997.

[30] L. Li. *Supply chain management: concepts, techniques and practices*. World Scientific Publishing Co., 2007.

[31] S.-H. Liao, C.-L. Hsieh, and Y. Lin. A multi-objective evolutionary optimization approach for an integrated location-inventory distribution network problem under vendor-managed inventory systems. *Annals of Operations Research*, 186, 2010.

[32] S. logistics. The bullwhip effect. `http://sinaslogisticsblog.blogspot.mx/2010/04/bullwhip-effect.html`.

[33] M. López-Ibanẽz, J. Dubois-Lacoste, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical report, Univiersité libre de Bruxelles, Belgium, 2011.

[34] E. Mastrocinque, B. Yuce, A. Lambiase, and M. Packianather. A multiobjective optimization for supply chain network using bees algorithm. *International Journal of Engineering Business Management*, 2013.

[35] E. Mezura-Montes and B. Hernández-Ocaña. Modified bacterial foraging optimization for engineering design. In *Proc. Artificial Neural Networks in Engineering Conference (ANNIE 2009)*, volume 19. ASME Press, 2009.

[36] E. Mezura-Montes, E. A. Portilla-Flores, and B. Hernández-Ocaña. Optimum synthesis of a four-bar mechanism using the modified bacterial foraging algorithm. *International Journal of Systems Science*, pages 1–21, 2012.

[37] E. Mezura-Montes, E. A. Portilla-Flores, and B. Hernández Ocaña. Optimum systhesis of a four bar mechanism using the modified bacterial foraging algorithm. *International Journal of Systems Science*, 2013.

[38] T. Moyaux, B. Chaiddraa, and S. DAmours. Information sharing as a coordination mechanism for reducing the bullwhip effect en a supply chain. *Journal IEEE Trans. on Systems, Man, and Cybernetics*, 2007.

[39] T. Moyaux and P. Mcburney. Reduction of the bullwhip effect in supply chains through speculation. Symp. Artificial Economics 2006, Lecture Notes in Economics and Mathematical Systems, 2006.

[40] T. Odonell. Minimising the bullwhip effect in a supply chain using genetic algorithms. *International Journal of Production Research*, 44, 2006.

[41] K. M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, pages 52–67, 2002.

[42] PE-Energy. The oilrig supply chain. `http://pe-energy.com/the-oilrig-supply-chain/`, 2012.

[43] M. Phelan and S. McGarraghy. Mitigating the bullwhip effect in supply chains using grammatical evolution. Pend, 2007.

[44] T. Pinto-Varela, A. P. F. D. Barbosa-Povoa, and A. Novais. Supply chain network optimization with enviromental impacts. 2nd International Conference on Engineering Optimization, Lisboa, Portugal, 2010.

[45] M. S. Pishavaee and J. Razmi. Enviromental supply chain network design using multiobjective fuzzy mathematical programming. *Applied Mathematical Modelling*, 36, 2012.

[46] P. Siarry and Z. Michalewicz. *Advances in Metaheuristics for Hard Optimization*. Springer, 2008.

[47] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimzation*, pages 431–359, 1997.

[48] F. Strozzi, N. Carlo, and J.-M. Zaldívar. Stability control in a supply chain: Total costs and bullwhip effect reduction. *The open operational research journal*, 1(2):51–59, 2008.

[49] F. Strozzi, C. Noe, and J.-M. Zaldivar. Stability control in a supply chain: Total costs and bullwhip effect reduction. *Open operational research*, 2008.

[50] H. Wang and B. He. Research on the reducing measures of bullwhip effect. International Conference on Software and Computer Applications IPCSIT, 2011.

[51] S. Zapotecas Martínez. Optimización multiobjetivo mediante un algoritmo híbrido basado en cómputo evolutivo y métodos clásicos de optimización. Master's thesis, CINVESTAV IPN, DF, México, 2007.

[52] Q. Zhang and H. Li. Moea/d: A multi-objective evolutionary algorithm based on decomposition. volume 11, pages 712–731, 2007.

[53] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical report, 2001.