UNIVERSIDAD DEL NORTE

ACREDITADA INSTITUCIONALMENTE
POR SU EXCELENCIA ACADEMICA

METAHEURÍSTICA DE OPTIMIZACIÓN COMBINATORIA PARA APROXIMAR SOLUCIONES AL PROBLEMA DEL AGENTE VIAJERO MULTI-OBJETIVO MEDIANTE CLUSTER DE CIUDADES

JESÚS DARÍO PEÑA SEGURA

UNIVERSIDAD DEL NORTE
DIVISIÓN DE INGENIERÍAS
MAESTRÍA EN INGENIERIA INDUSTRIAL
BARRANQUILLA
2017

METAHEURÍSTICA DE OPTIMIZACIÓN COMBINATORIA PARA APROXIMAR SOLUCIONES AL PROBLEMA DEL AGENTE VIAJERO MULTI-OBJETIVO MEDIANTE CLUSTER DE CIUDADES

JESUS DARÍO PEÑA SEGURA

TESIS DE GRADO

Presentado como requisito de grado para optar al título de Magister en Ingeniería Industrial

Director
MSc. Ing. Esneyder González Ponzón

Co-Director
PhD. Elías David Niño Ruíz

UNIVERSIDAD DEL NORTE
DIVISIÓN DE INGENIERÍAS
MAESTRÍA EN INGENIERIA INDUSTRIAL
BARRANQUILLA
2017

Nota de aceptación

_____

_____

_____
MSc. Esneyder González Ponzón
Director del proyecto

_____
PhD. Elías David Niño Ruíz
Co-Director del proyecto

Julio 5 de 2017

“Los que sembraron con lágrimas, con regocijo segarán”
Salmos 126:5

AGRADECIMIENTOS

Primeramente, quiero agradecer a Dios por darme la fortaleza y la sabiduría para realizar esta investigación.

En segundo lugar, agradezco a mi familia por ser un apoyo constante en cada una de las metas que me he propuesto, y por motivarme a seguir adelante en la obtención de este título de maestría.

También quiero agradecer a todo el equipo académico que hizo parte de este trabajo; gracias a mis directores de tesis, los ingenieros Esneyder González y Elías Niño, fueron una pieza clave para desarrollo de este proyecto, su confianza en mí, me motivó a dar lo mejor. Asimismo, a los profesores que hicieron parte del jurado, el Ingeniero Carlos Paternina y Héctor López, sus aportes enriquecieron esta investigación.

Finalmente quiero agradecer a la ingeniera Guisselle García y a la ingeniera Carmen Berdugo, porque sus consejos sabios permitieron que pudiera enfocarme de manera efectiva en este trabajo.

RESUMEN

La presente investigación propone una estrategia para la solución de problemas combinatorios. En particular, para el Problema del Agente Viajero Multi-Objetivo. En este sentido, se presenta una metaheurística de optimización combinatoria basada en Búsqueda Tabú, el Método del Vecino más Cercano y el cluster de ciudades a través del algoritmo K-Medoids. Para efectos prácticos, le llamaremos TS-KNN (por sus siglas en inglés: -Tabu Search using K-medoids and Nearest Neighbor).

Este algoritmo va a permitir un avance en el campo de la optimización combinatoria, debido a que, la estrategia de cluster de ciudades, no ha sido trabajada para el Problema del Agente Viajero Multi-Objetivo. Asimismo, será de gran interés para la comunidad científica y académica, ya que la literatura cataloga al TSP (Traveling Salesman Problem), como uno de los temas de investigación más tratados dentro de la rama de la optimización combinatoria, debido a su complejidad de solución, y a que muchos problemas en la industria y en la ciencia pueden ser reducidos al Problema del Agente Viajero.

Con el fin de validar la calidad de las soluciones, se tendrán en cuenta algunas instancias KRO (por las iniciales del primer autor) presentadas por Krolak, Felts, y Marble (1971) y que están disponibles en la librería de soluciones para el Problema del Agente viajero, TSPLIB (En inglés: *Library of sample instances for the Traveling Salesman Problem*), y se compararán los resultados con soluciones generadas por otros algoritmos documentados en la literatura científica.

Esta tesis, se desarrolla de a siguiente manera: en el capítulo 1, se presentan las generalidades del proyecto, como son; el planteamiento del problema, los objetivos, los alcances, consideraciones y limitaciones, y la metodología del proyecto. Seguidamente está el capítulo 2, en el cual se presenta el marco de teórico y el estado del arte. El capítulo 3, presenta el diseño e implementación de la metaheurística desarrollada. En el capítulo 4, se muestra un análisis de los resultados obtenidos con el algoritmo propuesto y se hace una comparación con los resultados arrojados por otros algoritmos disponibles en la literatura. Finalmente, se presentan las conclusiones y trabajos futuros con respecto al tema de esta investigación.

Tabla de contenido

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ALGORITMOS

1. GENERALIDADES

En este capítulo se presentan los principales planteamientos básicos a tener en cuenta para conocer el problema de investigación y la propuesta de solución. Asimismo, se plantean los lineamientos y directrices que enmarcan el desarrollo de este proyecto.

1.1. Formulación del Problema

En la actualidad, el desarrollo de herramientas de optimización para encontrar soluciones a problemas combinatorios, ha mantenido una fuerte relevancia, pese a los avances que se han conseguido en el área. Los problemas combinatorios se pueden modelar desde diferentes campos y aplicaciones, así como; el diseño de redes de comunicación, máquinas de visión, programación de horarios de la tripulación de aerolíneas, planeación corporativa, diseño asistido por computadoras, asignación de frecuencias de teléfonos celulares, manufactura, diseño de base de datos de información, biología computacional, entre otros. Además, pueden encontrarse en diversas áreas como, la programación lineal, teoría de grafos, inteligencia artificial y teoría de números (Du & Pardalos, 1998).

Uno de los problemas de optimización combinatoria más representativos, es el Problema del Agente Viajero (TSP por sus siglas en inglés: Traveling Salesman Problem), este consiste en determinar la ruta de menor distancia, que debería tomar un vendedor, que va a visitar *n* ciudades, pasando solo una vez por cada una de ellas, y regresando al mismo lugar del que partió. Este problema es motivo de diversos estudios, porque presenta la particularidad de que sus soluciones aumentan de manera exponencial con respecto al número de ciudades, y se ha demostrado que es un problema NP-Completo. Lo interesante, es que existen diversas aplicaciones en la ingeniería, que pueden ser mejoradas a través de metaheurísticas usadas para el TSP, por ende, optimizar este problema va a repercutir en posibilidades de mejoras en otras áreas de la industria.

En particular, este proyecto nace por la necesidad de aproximar mejores soluciones al problema del agente viajero multi-objetivo, a través de una estrategia de cluster de ciudades, la cual, nos va permitir aplicar algoritmos de optimización dentro de cada uno de las agrupaciones de ciudades, así como también, por fuera de los clusters para dar un mejor resultado en la ruta final.

Una consideración a tener en cuenta en nuestro problema, es el número de ciudades. Como sabemos, este es un aspecto de total relevancia por el nivel de complejidad del problema y porque de acuerdo con esto, la metaheurística planteada tendrá mayor impacto en la comunidad académica y científica. En este sentido, definimos como parámetros del problema 100 ciudades, cuyos datos de entrada serán tomados de las instancias KRO presentadas en la librería de soluciones TSPLIB.

Obtener soluciones eficaces y eficientes para el problema planteado, nos garantizará un avance en las técnicas de optimización que hasta ahora se han desarrollado. Asimismo, por aplicabilidad del problema del agente viajero a diversos contextos, aumentará los alcances de la optimización combinatoria en distintos campos científicos e industriales.

Basados en lo anterior, surge esta investigación que busca responder a la pregunta de **¿Es posible mejorar las aproximaciones a la solución del problema del agente viajero multi-objetivo, a través de una metaheurística de optimización combinatoria usando cluster de ciudades?**

1.2. Objetivos

1.2.1. Objetivo General

Diseñar e implementar una metaheurística de optimización combinatoria para aproximar soluciones al problema del agente viajero multi-objetivo a través del clúster de ciudades.

1.2.2. Objetivos Específicos

- Documentar los referentes teóricos y el estado del arte del problema del agente viajero.

- Diseñar e implementar un algoritmo metaheurístico que aproxime soluciones al problema del agente viajero, mediante el clusters de ciudades.

- Comparar los resultados con las aproximaciones dadas por algoritmos de optimización de gran impacto para el problema del Agente Viajero.

1.3. Alcances, Consideraciones y Limitaciones del Proyecto

- En este proyecto, se ha desarrollado una herramienta computacional que recibe dos matrices de pesos de las dos funciones objetivos. Los datos deben estar en formato txt separados por espacios. Se puede realizar la lectura con datos en formato Xls, sin embargo, el usuario debe modificar el código para este fin.

- El código se desarrolló en Matlab. Por tanto, para poder ejecutarlo se necesita la debida licencia.

- Los datos de las matrices de pesos de las funciones objetivos, las cuales pueden ser unidades de distancia, de costo, de tiempo, entre otros, deben estar normalizados para que las soluciones presentadas tengan sentido lógico.

- La metaheurística presenta las soluciones para el problema del agente viajero con dos objetivos, así como el respectivo frente de Pareto para los datos analizados. El usuario debe analizar el frente de Pareto y determinar la solución que más le convenga de acuerdo al contexto de su problema.

1.4. Metodología

El proceso metodológico de la tesis está compuesto por cuatro etapas.

Etapa 1. Consiste en definir el área de estudio en la que se enfocará el trabajo de tesis. Se procederá a realizar una revisión exhaustiva del estado del arte del problema del agente viajero, con el fin de generar una solución innovadora a dicho problema que nos permita tener soluciones eficientes y eficaces.

Etapa 2. En esta etapa se pretende diseñar el algoritmo metaheurístico que permita generar soluciones al problema del agente viajero con distancias simétricas, teniendo en cuenta búsqueda Tabú, el cluster de ciudades y la técnica del vecino más cercano. En el diseño de esta metaheurística, se tendrá en cuenta el proceso de selección del lenguaje de programación a utilizar, así como también las técnicas y métodos que permitirán la eficiencia del proceso.

Etapa 3. Validar la metodología propuesta a través del análisis comparativo de resultados, teniendo en cuenta técnicas de optimización combinatoria.

Etapa 4. En esta etapa se procederá a presentar los resultados obtenidos de la investigación a través del documento de tesis y de un artículo de investigación.

2. MARCO TEÓRICO

La evolución de la tecnología y la industria, a raíz de las exigencias del mercado y del entorno, ha permitido que las técnicas de mejora continua lleguen a todas las áreas científicas y también empresariales, esto ha impulsado el avance en técnicas de optimización para lograr mejores soluciones a los problemas a los que se enfrenta la industria. Las técnicas de optimización lineal y no lineal permitieron llegar a soluciones más precisas a los problemas de la industria, sin embargo, debido a los avances en la tecnología y a la globalización, los problemas se hicieron más complejos, lo cual le dio paso a nuevas técnicas de optimización, que permitieron seguir el proceso mejoramiento, el cual continúa de manera permanente, y exige el desarrollo constante de las modelos de optimización.

2.1. Marco Conceptual

A continuación, se presentan los conceptos fundamentales que enmarcan el tema de investigación.

2.1.1. Optimización

Deb (2001) define la optimización, como el proceso de encontrar una o más soluciones factibles, las cuales corresponden a valores extremos de uno o más objetivos. La necesidad de encontrar valores óptimos, viene del propósito conseguir soluciones que permitan la minimización de costos de producción o maximizar utilidades u otros objetivos.

La solución óptima, es aquella que cumple con las restricciones propuestas por el problema y adicionalmente representa la mejor solución de todas las posibles soluciones factibles. Las soluciones factibles son todas aquellas soluciones que cumplen con las restricciones impuestas por el problema. Consecuentemente, la factibilidad excluye a todas las soluciones que no cumplen con las restricciones propuestas por el problema, a este conjunto de soluciones se les conoce como soluciones no factibles (Niño, 2012). Esto puede apreciarse mejor en el grafico siguiente.

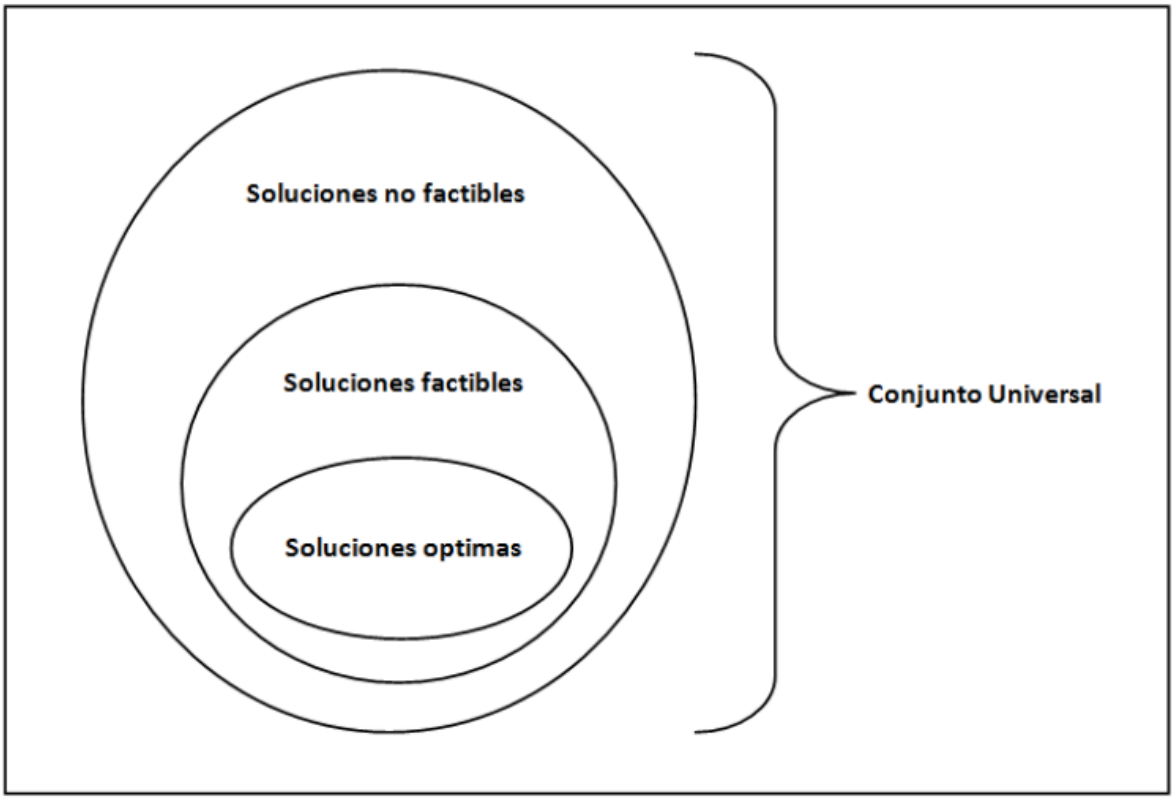

Figura 1. Conjunto Universal, Soluciones Factibles, No Factibles y Óptimas.

Fuente: Niño (2012)

Para muchas empresas, la optimización es un proceso fundamental, debido a que garantiza la estabilidad económica asociada a la minimización de costos con respecto a la mano de obra, a la materia prima, al consumo de energía, entre otros. En sectores productivos que dependen de recursos no renovables tales como el carbón, gas o el petróleo, La incorrecta planificación y ejecución de procesos no solo acarreará consecuencias negativas en el margen de utilidad sino también en el medio ambiente.

En términos matemáticos, el proceso de optimización se refiere a encontrar un valor óptimo x^* tal que $f(x^*)$ sea óptima, es decir, sea máxima o mínima. Por lo tanto, optimizar se refiere a maximizar (*max*) o minimizar (*min*) una función conocida como función objetivo (Niño, 2012). El modelo general es definido de la siguiente manera:

$$min\ f(x) | x \in \Omega$$

En donde $\Omega \subseteq R^n$ y $f: \Omega \to R$. Ω es la región factible del problema.
Una propiedad importante es que:

$$min\ f(x) = -\max\ -f(x) \text{ y } max\ f(x) = -min - f(x)$$

Este factor se da porque las funciones $f(x)\ y\ -f(x)$ son funciones opuestas

Esta propiedad es ventajosa, al momento de utilizar herramientas computacionales para implementar métodos de aproximación numérica que nos permitan tener una idea de cuál es la solución óptima del problema. Por lo tanto, es suficiente la implementación para uno de los dos casos y dado el evento de requerir el opuesto, simplemente se multiplica por −1 la función objetivo. Es importante hacer énfasis en que aun cuando la solución exacta es la primordial y deseada, muchas veces esta no es fácil de calcular o simplemente es imposible de hacerlo (Niño, 2012).

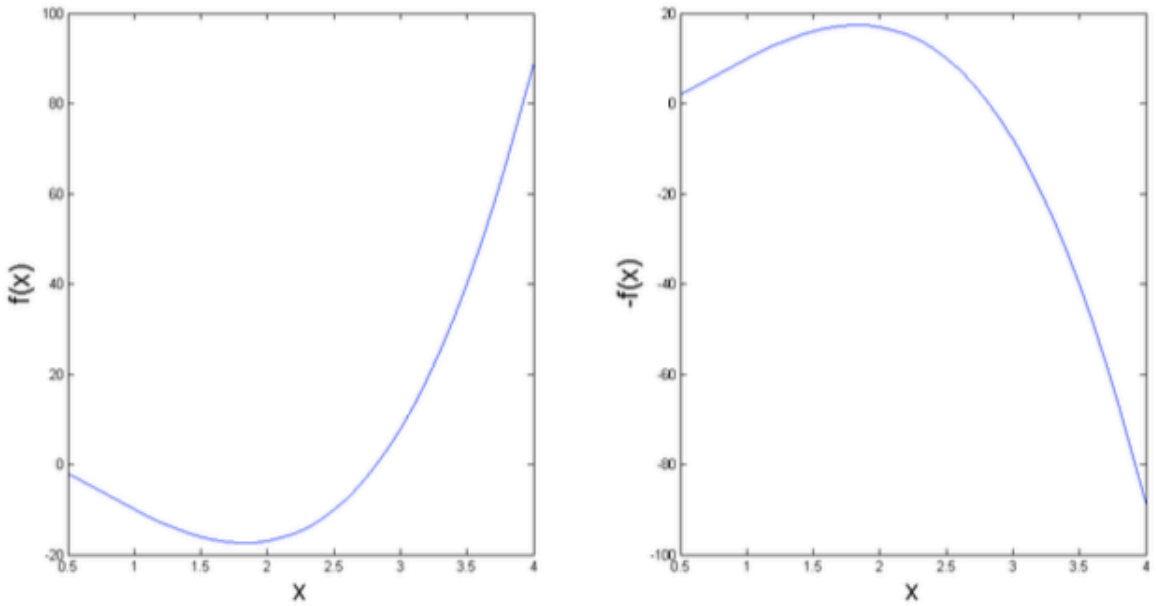

Figura 2. Gráficas de $f(x)\ y\ -f(x)$, donde se evidencia que son opuestas entre sí.

2.1.2. Optimización Mono-objetivo

Niño (2012), define la optimización mono-objetivo como el proceso de optimizar una sola función teniendo en cuenta una serie de restricciones que están basadas en restricciones del mundo real. Matemáticamente, un problema de optimización mono-objetivo es expresado de la siguiente manera:

$$Opt\ \{max,\ min\} \quad f(X)$$

Sujeto a:

$$H(X) = 0$$
$$G(X) \leq 0$$
$$X_l \leq X \leq X_u$$

En Donde *X* es un vector de n variables, $f(X)$ es la función objetivo, $H(X)$ y $G(X)$ son las restricciones del problema y X_l , X_u son los límites para *X.*

2.1.3. Optimización Multi-Objetivo

En la vida real, cuando se trata de solucionar un problema, muchas veces interesa optimizar más de una función. Debido a esto, nacen los problemas de optimización multi-objetivo. La diferencia con el modelo mono-objetivo radica únicamente en la función objetivo, ya que en este nuevo modelo se plantea un conjunto de funciones a optimizar. Por lo tanto, la función objetivo de un problema multi-objetivo se define de la siguiente manera:

$$Opt\left\{max_i, min_j,\right\} \quad F(X) = \{f_1(X), f_2(X), \dots, f_m(X)\}$$

Sujeto a:

$$H(X) = 0$$
$$G(X) \leq 0$$
$$X_l \leq X \leq X_u$$

En Donde F(*X)* es un vector de m funciones objetivos, $H(X)$ y $G(X)$ son las restricciones del problema, X_l , X_u son los límites para *X y* m es el número de objetivos del problema. Los índices i y j indican que algunas funciones pueden ser maximizadas o minimizadas. Cuando existen casos en los cuales la mejora de una función implica a desmejora de otra, se reconoce con el nombre de conflicto de intereses *Niño (2012).*

La optimización multi-objetivo es muy diferente de la optimización de un problema mono-objetivo. En un problema mono-objetivo, se busca la mejor solución de todas y bajo este esquema, solo existe un punto de comparación por el cual se decide si una solución es mejor que otra. En la siguiente gráfica, se muestra un ejemplo en el que se ven tres soluciones para un problema particular. Si el objetivo del problema fuera maximizar, la solución sería f_2 por ser la menor del conjunto, en caso contrario, si el objetivo fuera la minimización de la función, la solución sería f_3.

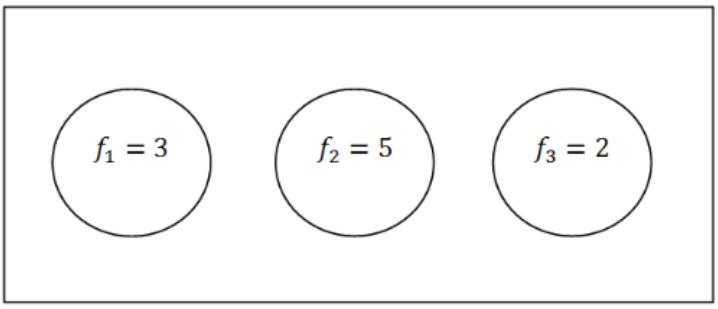

Figura 3. Tres soluciones para un problema particular mono-objetivo.

Fuente: Niño (2012)

En caso que el problema fuera multi-objetivo, no existe un unico punto de referencia por el cual se pueda decir que una solución es mejor que otra, por ejemplo, considere las soluciones que se presentan en la Figura 4, donde se muestran tres soluciones para un problema particular multi-objetivo. Si todos los objetivos se minimizaran, no se puede determinar entre las dos primera soluciones cual es mejor, ya que $f_{11} > f_{21}$ y $f_{12} < f_{22}$, sin embargo, las dos primeras soluciones son mejores que la tercera, debido a que en este caso ambas funciones son menores. Por lo tanto, se dice que la solución 3 es dominada por la solución 1 y la solución 2. Lo anterior nos lleva dar las siguientes definiciones:

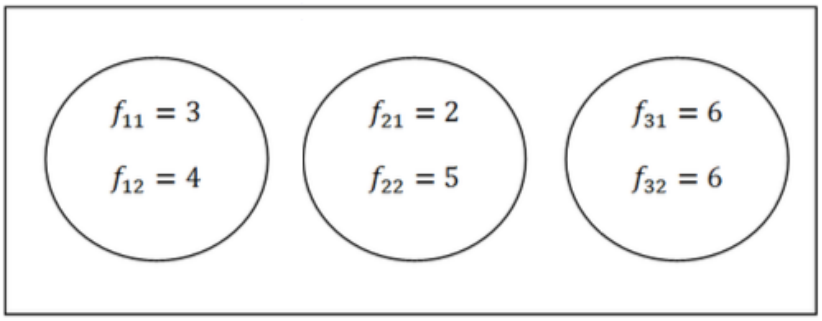

Figura 4. Tres soluciones para un problema particular multi-objetivo.

Fuente: Niño (2012)

Dominancia de Pareto: Dados dos vectores $X, Y \in R^n$ y $F(X) = \{F_1(X), F_2(X), \dots, F_n(X)\}$ se dice que X domina a Y, esto es: $F(X) \prec F(Y)$, si y solo si:

$$\forall i \in \{1, 2, \dots, m\}\ f_i(X) \leq f_i(Y)\ \ \wedge \exists j \in \{1, 2, \dots, m\}\ f_i(X) < f_j(Y)$$

Optimalidad de Pareto: Se dice que una solución es óptima si y solo si para $X \in S \subseteq R^n, n \geq 2$ y $F(X) = \{F_1(X), F_2(X), \ldots, F_n(X)\}$ se cumple que:

$$\nexists X \in S / F(X) \prec F(X^*)$$

Frente de Pareto Es el conjunto de soluciones no dominadas. La dimensión de este dependerá exclusivamente del número de funciones objetivo del problema, es decir R^m, en donde *m* es el número de funciones objetivo del problema. Un ejemplo en R^2 y R^3 puede apreciarse en la figura 5.

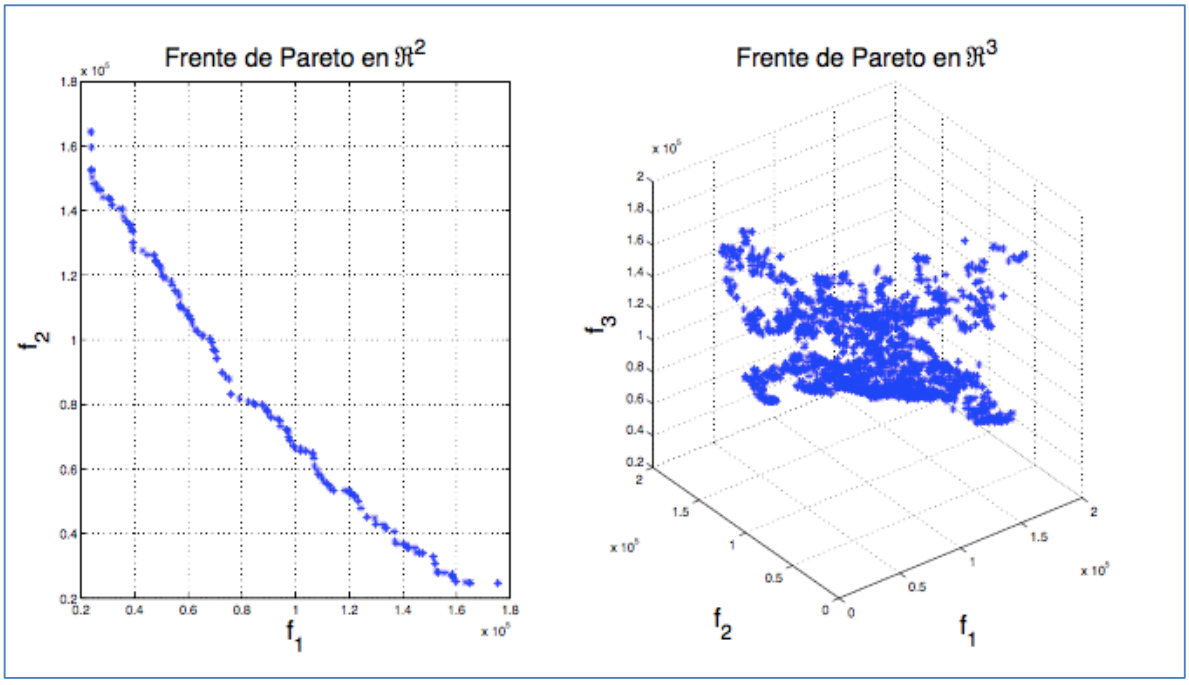

Figura 5. Frente de Pareto para R^2 y R^3

Fuente: Niño (2012)

Conjunto Convexo: Un conjunto *S* es convexo si y solo si: $\forall x, y \in S, \forall \alpha [0,1]$ se cumple que:

$$\alpha x + (1 - \alpha) y \in S$$

Función Convexa: Una función es convexa si y solo si, $\forall\alpha[0,1]$ y $\forall x_1, x_2 \in R$ se cumple que:

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$$

Función no Convexa: Una función es no convexa si no cumple con $f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha)f(x_2)$. En otras palabras, existen combinaciones no lineales de x_1 y x_2 que no se encuentran contenidas dentro del rango de la función.

Finalmente, las técnicas las técnicas clásicas de optimización multi-objetivo, funcionan aproximando las soluciones del problema planteado mediante la optimización de un problema mono-objetivo. Una de las estrategias más conocida es la suma con pesos.

2.1.3.1. Suma con pesos

Niño (2012), define la suma con pesos como un método donde se crea un modelo mono-objetivo mediante la ponderación de pesos α_i en las *m* funciones objetivo del problema. Matemáticamente, se puede definir de la siguiente forma:

$$Opt\left\{max,\ min\right\}\ F(X) = \sum_{i=1}^{m} \alpha_i f_i(x)$$

Sujeto a:

$$H(X) = 0$$
$$G(X) \leq 0$$
$$X_l \leq X \leq X_u$$
$$\sum_{i=1}^{m} \alpha_i = 1$$
$$\alpha_i \in [0,1]$$

De acuerdo con lo anterior, se puede ver que $F(X)$ se crea como la combinación lineal de $\{f_i(X)\}_{i=1}^{m}$, es decir, de las funciones objetivos del problema. De esta manera cada vez que cambie el valor de los $\{\alpha_i\}_{i=1}^{m}$ se obtendrá una nueva solución óptima. El conjunto de soluciones óptimas, hacen parte del Frente de Pareto del problema.

A continuación, se presenta el método de la suma con pesos descrito a través de su algoritmo.

Algoritmo 1. Método de la Suma con Pesos para un problema de Minimización Bi-objetivo.

Parámetros: $[a, b]$, $f_1(x), f_2(x) \in C[a, b]$, δ, λ
{$[a, b]$ son los límites del intervalo}
{$f_1(x), f_2(x)$ son las funciones objetivos a ser optimizadas}
{δ es la presición con la que se desea obtener la solución}
{λ es el incremento del peso α}
Salida: Conjunto S con soluciones óptimas
1: $S \leftarrow \phi$
2: **para** $\alpha = 0$ hasta 1 **haga**
3: $F(x) \leftarrow \alpha f_1(x) + (1 - \alpha) f_2(x)$
4: $x^* \leftarrow BusquedaDorada(F(x), a, b, \delta)$
5: $S \leftarrow S \cup \{(x^*, f_1(x^*), f_2(x^*))\}$
6: $\alpha \leftarrow \alpha + \lambda$
7: **fin para**
8: **regresa** S

Fuente: Niño (2012)

2.1.4. Optimización Combinatoria

La ingeniería y otras áreas enfrentan diariamente problemas cada vez más complejos que deben ser solucionados a partir de su expresión como problemas de optimización. Algunos no pueden ser solucionados con técnicas clásicas de programación lineal y otros, son de naturaleza no lineal. Sin embargo, dentro del conjunto de problemas, existen unos para el cual, a partir de ciertos parámetros, sus soluciones son difíciles de encontrar en un tiempo polinomial. Este tipo de problemas corresponde por lo general, a problemas de naturaleza combinatoria. La optimización combinatoria es un campo de las matemáticas aplicadas, que combina técnicas combinatorias de programación lineal y de teoría de algoritmos, para resolver problemas sobre estructuras discretas (Niño, 2012).

Papadimitriou y Steiglitz (2013), definen la optimización combinatoria como una rama de las ciencias matemáticas, que plantea la construcción de métodos que permitan realizar combinaciones y ordenamientos a grupos de elementos, es decir alteraciones de las posiciones de los elementos dentro del universo. El objetivo es que se puedan conseguir patrones adecuados que permitan resolver determinado tipo de problemas de complejidad NP o NP-Hard.

La optimización de un problema combinatorio, se relaciona con algunos criterios de costo, los cuales proveen una medida cuantitativa de la calidad de cada solución (Aarts y Lenstra, 1997).

Los problemas de decisión, hacen parte decisión hacen parte de esta rama de la optimización entre los cuales se destaca el Problema del Agente Viajero (Traveling Salesman Problem, TSP).

2.1.4.1. Problema Combinatorio

Los problemas combinatorios son problemas derivados del área de la optimización cuya representación se basa o puede ser reducida a estructuras discretas tales como vectores, matrices, listas o grafos. Generalmente el espacio de soluciones factibles asociadas a este tipo de problemas es exponencial, por lo tanto, a medida que crece el número de variables de decisión del problema, el problema se vuelve intratable, dificultando encontrar la solución óptima.

Algunos problemas combinatorios pertenecen a la categoría conocida con el nombre de Nondeterministic Polynomial Time (NP). En NP se encuentran todos los problemas cuya solución óptima puede ser obtenida en tiempo polinómico por una máquina de Turing no determinista (Niño, 2012). Es decir, que la solución óptima se puede aproximar a través de la implementación de una estrategia probabilística. Por otra parte, están los problemas combinatorios conocidos como NP-Completo. Estos son del tipo de problemas que, para garantizar la obtención de una solución óptima, se debe realizar una búsqueda exhaustiva sobre el espacio de soluciones factibles, lo que posiblemente implicaría años para la solución del problema.

2.1.5. El Problema del Agente Viajero - TSP

El problema del agente viajero (TSP), consiste en que, dado un conjunto de ciudades, un viajero debe visitar todas las ciudades pasando por ellas sólo una vez, regresando al punto de origen y haciendo el menor tiempo posible (Papadimitriou, 1982).

Desde los métodos de Ramificación y Acotación hasta los basados en la Combinatoria Poliédrica, pasando por los procedimientos metaheurísticos, todos han sido inicialmente probados en el TSP, convirtiéndose éste, en una prueba obligada para “validar” cualquier técnica de resolución de problemas enteros o combinatorios. La librería de instancias y soluciones para el problema del agente viajero, TSPLIB (Reinelt, 1991), la cual es de domino público, contiene un conjunto de ejemplos del TSP con la mejor solución obtenida hasta la fecha y, en algunos casos, con la solución óptima para determinadas instancias (Cunquero, 2003).

2.1.6. Heurística y Meta-Heurística

Las técnicas heurísticas y meta-heurísticas, se refieren a una serie de métodos o algoritmos exploratorios que actúan durante la resolución de problemas cuyas soluciones se consiguen por la evaluación del progreso logrado en la búsqueda de un resultado final. Estas técnicas son empleadas en una gran cantidad de disciplinas y áreas del conocimiento y su finalidad es la de entregar soluciones que satisfagan al máximo el problema, Gallart (2009).

Estas técnicas, aunque no aseguran una solución óptima, permiten encontrar buenas soluciones utilizando razonablemente los recursos computacionales. Por lo cual, son muy usadas cuando no existe un método de solución exacto, que pueda optimizar haciendo un uso eficiente de los recursos.

2.1.7. El vecino más cercano

Uno de los métodos heurísticos más sencillos para el análisis del TSP es el llamado "método del vecino más cercano". Este tiene como objetivo, construir un ciclo Hamiltoniano, basándose en el vértice que se encuentre más cercano a uno dado, minimizando así, la distancia recorrida. Este algoritmo es debido a Rosenkrantz, Stearns y Lewis (1977).

El procedimiento es bastante intuitivo, y de manera práctica, cumple con el objetivo de generar una ruta tratando de minimizar el costo o la distancia de una ciudad a otra, sin embargo, existe una debilidad en este método, la cual se refiere a que, al escoger siempre la menor distancia entre las opciones inmediatas, es posible que rechace una mejor ruta basado en las distancias futuras. Esto es lo que se conoce como *miopía* del procedimiento, ya que, en una iteración escoge la mejor opción disponible sin "ver" que esto puede obligar a realizar malas elecciones en iteraciones posteriores (Cunquero, 2003).

2.1.8. Búsqueda Tabú

Niño (2012) describe la Búsqueda Tabú (Glover & Laguna, 1997) como una estrategia de búsqueda local que permite la solución de problemas combinatorios. El método se basa en la perturbación de soluciones y la vecindad de las mismas. Una perturbación consiste en alterar, cambiar o transformar una solución en otra. Esto dependerá de cómo se presentan las soluciones. Por lo tanto, una representación vectorial puede ser fácilmente perturbada intercambiando posiciones en el vector, una representación por medio de listas enlazadas puede ser fácilmente perturbada eliminando y/o agregando nodos a la lista y, por último, una representación binaria, puede ser fácilmente perturbada cambiando 1 por 0. Existen diversas formas de representar las soluciones, por lo tanto, la manera de perturbarla varía entre ellas. El vecindario de una solución s es $N(s)$ y está

relacionado con las soluciones que pueden ser alcanzadas con tan solo una perturbación. El vecindario de todas las soluciones representa el espacio de todas las soluciones factibles.

La estructura general de la Búsqueda Tabú se define de la siguiente manera:

- **Paso 1.** Generar una solución aleatoria *s*.
- **Paso 2.** Generar una solución s' del vecindario de s, $s' \leftarrow P(s), s' \in N(s)$
- **Paso 3.** Si $f(s') < f(s)$ hacer $s \leftarrow s'$. Verificar condición de parada. Ir al paso 2.

La Búsqueda Tabú solo sustituye la mejor solución cuando la encontrada supera a la actual en la función objetivo. Este tipo de estrategias, presenta la desventaja de que puede converger hacia óptimos locales.

Es posible que ocurran ciclos durante la exploración de vecindarios utilizando esta técnica, por lo tanto, para evitar ciclos algunos autores han propuesto la lista Tabú que permite recolectar las últimas *x* soluciones analizadas.

Por lo general el método termina cuando la función objetivo no ha mejorado por n iteraciones. Un esquema procedimental básico de la Búsqueda Tabú se presenta a continuación:

Algoritmo 2. Método de la Búsqueda Tabú

Parámetros: s, f, N
{s solución inicial.}
{f función objetivo.}
{N máximo número de iteraciones sin mejorar la función objetivo.}
Salida: s^* como óptimo local.
$n \leftarrow 1$ { contador de no mejoras de s en la función objetivo.}
haga
  $s' \leftarrow P(s)$
  si $f(s') < f(s)$ **entonces**
    $s \leftarrow s'$
    $n \leftarrow 0$ { se reinicia el contador de no mejoras.}
  sino
    $n \leftarrow n + 1$ { se incrementa el contador de no mejoras.}
  fin si
hasta $n > N$ { hasta que la función no haya mejorado N veces}
$s^* \leftarrow s$
regresa s^*

2.1.9. K-Medoids Clustering

Park y Jun (2009), explican el algoritmo K-Medoids Clustering, como un método cuyo objetivo es la partición o agrupamiento de datos para los cuales la media o la mediana no tienen una definición clara. Es un método más robusto que el k-means, cuando dentro del conjunto de datos existen elementos atípicos.

Es similar a k-means, y el objetivo de ambos métodos es dividir un conjunto de mediciones u observaciones en k subconjuntos o clusters de modo que los subconjuntos minimicen la suma de las distancias entre una medida y un centro de la medida del cluster. En el algoritmo k-means, el centro del subconjunto es la media de las mediciones en el subconjunto, a menudo llamado centroide. En el algoritmo k-medoids, el centro del subconjunto es un miembro del subconjunto, llamado medoid.

El algoritmo de k-medoids devuelve los medoids, que son puntos reales del conjunto de datos. Esto permite usar el algoritmo en situaciones donde la media de los datos no existe dentro del conjunto de datos. La principal diferencia entre k-medoids y k-means es que los centroides devueltos por k-means pueden no estar dentro del conjunto de datos. Por lo tanto, k-medoids es útil para agrupar datos categóricos donde una media es imposible de definir o interpretar.

La función k-medoids proporciona varios algoritmos iterativos que minimizan la suma de distancias de cada objeto al medoid de su respectivo cluster. Uno de los algoritmos se llama *Partitioning Around Medoid* (PAM) (Kaufman & Rousseeuw, 2009), que procede en dos pasos que se presentan a continuación:

Build-Step: Selecciona *k* de los *n* puntos como un medoid potencial. Luego se conforman los *k* clusters iniciales, asociando cada punto al medoid más cercano.

Swap-Step: Dentro de cada cluster, se prueba cada punto como un medoid potencial, verificando si la suma de las distancias del clúster disminuye. Si es así, el punto se define como un nuevo medoid y luego, se conforma un nuevo cluster con los puntos más cercanos a dicho medoid.

El algoritmo itera los pasos build-and-swap hasta que los medoids no cambian, o hasta que otros criterios de terminación se cumplen.

En la figura 3, podemos ver de manera gráfica el resultado del algoritmo K-Medoids, para un grupo de 100 datos generados de manera aleatoria, para los cuales se generaron dos clusters.

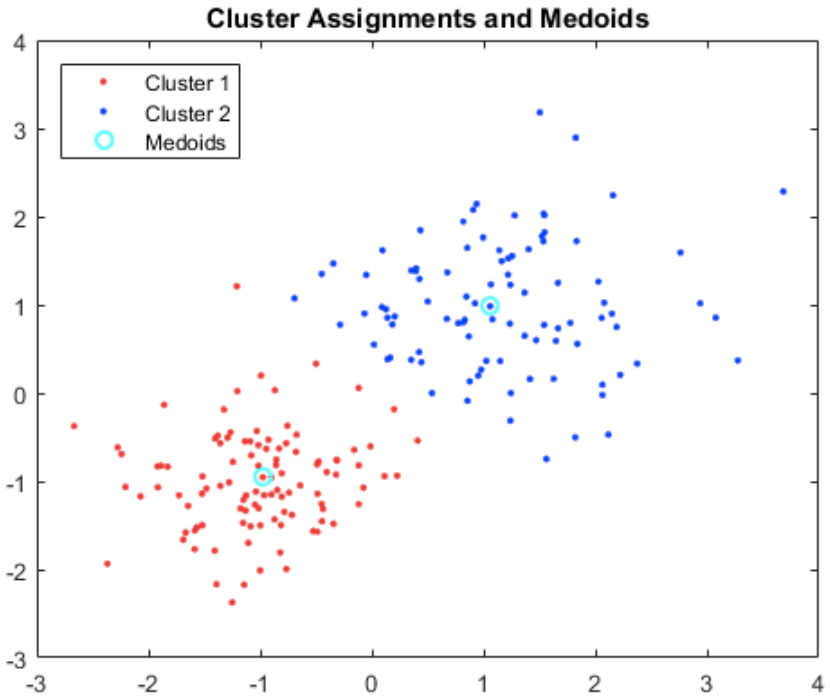

Figura 6. Agrupación de datos aleatorios en dos clusters usando K-Medoids
Fuente: Tomado de Web en: es.mathworks.com

2.1.10. Metricas de optimización multi-objetivo

Los problemas de optimización multi-objetivo como se dijo anteriormente, tienen el propósito de optimizar varios objetivos al mismo tiempo, sean estos de maximización, de minimización o la combinación de ambos. Lo anterior, implica que la solución de este tipo de problemas esté ligada a un conjunto de soluciones no dominadas (Niño, 2012). El conjunto de soluciones no dominadas tiene por nombre Frente de Pareto y representa una aproximación al conjunto de soluciones óptimas del problema.

Encontrar el conjunto de soluciones óptimas del problema es un trabajo que puede requerir años de investigación, sobre todo cuando el esquema de investigación multi-objetivo se basa en problemas de optimización combinatoria los cuales son en su mayoría NP completos (Niño, 2012). Por lo tanto, las metaheurísticas ofrecen una aproximación a lo que sería la solución óptima del problema. Sin embargo, ninguna garantiza obtener las soluciones óptimas del problema. Lo anterior implica, que se deban usar métricas de optimización multi-objetivo para inferir la calidad de las soluciones generadas por un algoritmo.

A continuación, se presentan algunas métricas de optimización multi-objetivo ofrecidas por la literatura, así como estrategias para realizar correctamente la experimentación.

2.1.10.1. Frente de Pareto conocido (PF_{Known})

Como se mencionó anteriormente, un frente de Pareto es el conjunto de soluciones óptimas para un problema de optimización multi-objetivo. Si es una metaheurística la que genera el frente de Pareto, entonces se le llama frente de Pareto conocido o $\boldsymbol{PF_{Known}}$ (por sus siglas en inglés) debido a que es una aproximación al conjunto de soluciones óptimas del problema, a menos que se demuestre la optimalidad de las soluciones encontradas. Por otra parte, el frente de pareto óptimo es conocido como frente de Pareto real $\boldsymbol{PF_{True}}$ (por sus siglas en inglés).

Garantizar la optimalidad de PF_{Known} cuando se trata de una metaheurística o algorimo puede ser una tarea dificil, sobre todo cuando el método de aproximación es probabilístico, por lo tanto es indispensable el uso de metricas que de acuerdo a ciertos indicadores nos den una idea de la calidad de las soluciones encontradas (Niño, 2012).

- **Frente de Pareto Ideal ($\boldsymbol{PF_{True}}$)**

Las Métricas de Optimización Multi-Objetivo son fórmulas matemáticas y/o estadísticas que permiten, mediante diversos análisis, determinar la calidad del conjunto de soluciones generadas por un algoritmo y la eficiencia computacional del algoritmo que genera dichas soluciones. Los análisis pueden llevarse a cabo mediante dos conjuntos de métricas: métricas de medición unitaria y métricas de medición grupal. Las métricas de medición unitaria utilizan únicamente el PF_{Known} para realizar los cálculos, con lo cual se puede estimar el número de soluciones generadas, así como la distribución de soluciones en el espacio. Por otra parte, las técnicas de medición grupal utilizan dos o más frentes de Pareto con el fin de brindar información acerca del grado de proximidad y convergencia de las soluciones propuestas. Para el cálculo de este conjunto se requiere, por lo menos, un PF_{Known} y el PF_{True}. También, es posible llevar a cabo comparaciones entre diversos conjuntos factibles y determinar cual de ellos ofrece una mejor aproximación tomando como referencia al PF_{True}. Debido a esto, un completo análisis de las soluciones generadas a partir de un algoritmo implica combinar métricas de ambos tipos, (Niño, 2012). Sin embargo, debido a la complejidad de los problemas tratados en la rama de optimización combinatoria, muchas veces no se cuenta con PF_{True} , por lo cual, se debe contruir PF_{True} partiendo de la información existente. En este sentido, se puden dar dos casos:

(i) Dos o más Frentes de Pareto en el Análisis. Si se tiene dos o más frentes de Pareto, se prosigue de la siguiente manera:

Paso 1. Dados los frentes de Pareto $PF_1, PF_2, \dots, PF_q$ de diversos algoritmos $A_1, A_2, \dots, A_q$ donde *q* es el número de algoritmos, se crea un nuevo conjunto mediante la unión de los frentes, esto es:

$$\boldsymbol{PF_A = \bigcup_{i=1}^{q} PF_i}$$

Paso 2. Se retiran las soluciones dominadas de PF_A. Se hace $PF_{True} \longleftarrow PF_A$
Un aspecto importante de toda la prueba experimental, es que los frentes de Pareto sean obtenidos bajo las mismas condiciones computacionales, teniendo en cuenta que:

a. Todos los algoritmos deben ser ejecutados igual número de veces. Debido a que las metaheurísticas son estrategias probabilísticas, cada algoritmo debe ser ejecutado por lo menos dos veces para así tomar las soluciones no dominadas de las dos corridas. Este conjunto final no dominado, representa el conjunto de soluciones no dominadas PF_{Known} del algoritmo. De manera similar, se lleva a cabo la construcción del frente PF_{Known} para *n* corridas de un algoritmo.

b. Todos los algoritmos deben ser implementados bajo el mismo lenguaje computacional, utilizando igual capacidad de cómputo y bajo el mismo sistema operativo.

(ii) Un solo frente de Pareto en el Análisis. Cuando se tiene un solo frente de Pareto, determinar la aproximación al Frente de Pareto Ideal es una tarea casi imposible, a menos que, mediante estrategias analíticas se demuestre la convergencia del método. Sin embargo, numéricamente, es posible determinar la solución local hacia la cual converge el método. Para esto se deben llevar a cabo los siguientes pasos:

Paso 1. Se realizan n corridas del algoritmo. En cada prueba se guarda el PF_i encontrado.

Paso 2. Se construye un nuevo frente temporal (PF_T) con la unión de todos los frentes de Pareto obtenidos en las *n* corridas del algoritmo.

$$PF_T = \bigcup_{i=1}^{n} PF_i$$

Paso 3. Se remueven las soluciones dominadas de PF_T se hace $PF_{True} \leftarrow PF_T$.

Paso 4. Finalmente, se realizan las respectivas comparaciones de los PF_{Known} de cada corrida (PF_i) con PF_{True}

2.1.10.2. Generación de Vectores No Dominados (GNDV)

La métrica de Generación de Vectores No Dominados consiste en contar el número de elementos (Soluciones) de un Frente de Pareto PF_{Known}, esto es:

$$GNDV = \|PF_{Known}\|$$

Como puede apreciarse, esta métrica requiere únicamente de PF_{Known} para ser calculada. Un aspecto importante al momento de analizar un algoritmo por medio de esta métrica es la conveniencia de su valor. Es preferible tener pocas soluciones con una buena calidad a tener muchas soluciones de poca calidad. Por esto, es importante el uso de otra métrica para poder llevar a cabo un juicio correcto sobre un algoritmo al momento de utilizar *GNDV* (Niño, 2012).

2.1.10.3. Proporción de generación de Vectores No Dominados (*RGNDV*)

Mide la Proporción de Generación de Vectores No dominados (*RGNDV*) mide la proporción de la cantidad de soluciones no dominadas generadas por un algoritmo y la cantidad de soluciones que contiene el Frente de Pareto Ideal (Niño, 2012).

$$RGNDV = \left(\frac{GNDV}{PF_{True}}\right) * 100\% = \left(\frac{GNDV}{GNDV_{True}}\right) * 100\%$$

En caso que *RGNDV* sea igual a 100%, solo puede inferirse que la cantidad PF_{Known} es igual a PF_{True}. Esto no significa que sean las mismas soluciones, ni tampoco que necesariamente las soluciones convergen a PF_{True}.

2.1.10.4. Generación Real de Vectores No Dominados (*ReGNDV*)

Al momento de medir el rendimiento de un algoritmo, la métrica de Generación Real de Vectores No Dominados, es una de las más importantes. Consiste en contar el número de soluciones PF_{Known} que se encuentran en PF_{True}. Con esto se obtiene un estimado del número de soluciones generadas que convergieron hacia PF_{True} (Niño, 2012). Matemáticamente, se define de la siguiente manera:

$$ReGNDV = \sum_{i=1}^{\|PF_{Known}\|} \beta_i$$

En donde β_i es definida como:

$$|\beta_i| = \begin{cases} 1, & s_i \in PF_{Known} \wedge s_i \in PF_{True} \\ 0, & s_i \in PF_{Known} \wedge s_i \notin PF_{True} \end{cases}$$

En donde s_i representa la solución *i-ésima* del conjunto PF_{Known}. De esta manera vemos que, la métrica $ReGNDV$ a través de la función β_i solo cuenta las soluciones de PF_{Known} que están en PF_{True}.

2.1.10.5. Distancia Generacional (*GD*)

La métrica de la Distancia Generacional (Van Veldhuizen & Lamont, 2000) se usa para determinar en promedio que tan lejos está PF_{Known} de PF_{True}. Su formulación matemática se presenta a continuación:

$$GD = \left(\frac{1}{\|PF_{known}\|}\right)\left(\sum_{i=1}^{\|PF_{Known}\|} (d_i)^m\right)^{(1/m)}$$

En donde *m* es el número de objetivos del problema del cual proviene PF_{Known}.
d_i es la mínima distancia de cada miembro $s_i \in PF_{Known}$ con el miembro más cercano del conjunto de soluciones PF_{True}, es decir:

$$d_i = min\left\{\|s_i - x_1\|_F, \|s_i - x_2\|_F, \ldots, \left\|s_i - x_{\|PF_{True}\|}\right\|_F\right\}$$

En donde $x_i \in PF_{True}$ y $\|.\|_F$ es la norma Frobenius o Euclideana:

$$\|x\|_F = \sqrt{\sum_{i=1}^{n} (x_i)^2}$$

Por lo tanto, es necesario al utilizar esta métrica es necesario contrastar por medio de la norma Frobenius cada elemento del conjunto PF_{Known} con todos los elementos del conjunto PF_{True} para determinar cuál es el más cercano.

A continuación, se presenta el algoritmo que muestra el conjunto de pasos para calcular la métrica *GD* a un conjunto PF_{Known} y un conjunto de referencia PF_{True}. La complejidad de este método en el peor de los casos es $O(n^2)$ (Niño, 2012).

Algoritmo 3. Métrica de la Distancia Generacional (*GD*).

Parámetros: $PF_{known} \in \Re^{n \times m}, PF_{true} \in \Re^{q \times m}$
$\{PF_{known}$ Frente de Pareto conocido (aproximado)$\}$
$\{PF_{true}$ Frente de Pareto Real$\}$
Salida: Valor S con la Distancia Generacional del conjunto PF_{known} con respecto a PF_{true}

$S \leftarrow 0$
para $i = 1$ **hasta** $\|PF_{known}\|$ **haga**
  $s_i \leftarrow PF_{known}(i)$
  $d_i \leftarrow -1$
  para $j = 1$ **hasta** $\|PF_{true}\|$ **haga**
    $s_j \leftarrow PF_{true}(j)$
    $d \leftarrow \|si - sj\|_F$
    si $d < d_i \vee d_i = -1$ **entonces**
      $d_i \leftarrow d$
    fin si
  fin para
  $S \leftarrow S + d_i^m$
fin para
$S \leftarrow \frac{\sqrt[m]{S}}{\|PF_{known}\|}$
regresa S

2.1.10.6. Distancia Generacional Inversa (IGD)

La distancia generacional inversa o *IGD* (por sus siglas en inglés) es utilizada para medir la distancia mínima entre PF_{True} y PF_{Known}. Su formulación matemática es como sigue:

$$IGD = \left(\frac{1}{\|PF_{true}\|}\right)\left(\sum_{i=1}^{\|PF_{true}\|} \tilde{d}_i\right)$$

En donde $\tilde{d}_i$ es la mínima distancia de cada miembro $x_i \in PF_{True}$ con el miembro más cercano del conjunto de soluciones PF_{Known}, es decir:

$$\tilde{d}_i = min\left\{\|x_i - s_1\|_F, \|x_i - s_2\|_F, \dots, \left\|x_i - s_{\|PF_{known}\|}\right\|_F\right\}$$

En donde $s_i \in PF_{Known}$ y $\|.\|_F$ es la norma Frobenius.

$$\|x\|_F = \sqrt{\sum_{i=1}^{n} (x_i)^2}$$

Un valor de 0 en esta métrica indica que $PF_{known} = PF_{true}$. Un valor diferente de 0 puede significar dos cosas. La primera es que el número de puntos PF_{true} es mayor al número de puntos PF_{known}, o que algunas soluciones PF_{true} domina a las soluciones PF_{known}, por lo tanto, será necesario apoyarse en el valor de la métrica *GD* para llevar a cabo un juicio correcto sobre los resultados (Niño, 2012).
A continuación, se expresa la métrica *IGD* como pseudocódigo, la complejidad al igual que en el método anterior es $O(n^2)$.

Algoritmo 4. Métrica de la Distancia Generacional Inversa (*IGD*)

Parámetros: $PF_{known} \in \Re^{n \times m}, PF_{true} \in \Re^{q \times m}$
$\{PF_{known}$ Frente de Pareto conocido (aproximado)$\}$
$\{PF_{true}$ Frente de Pareto Real$\}$
Salida: Valor S con la Distancia Generacional Inversa del conjunto PF_{known} con respecto a PF_{true}

$S \leftarrow 0$
para $i = 1$ **hasta** $\|PF_{true}\|$ **haga**
  $s_i \leftarrow PF_{true}(i)$
  $\tilde{d}_i \leftarrow -1$
  para $j = 1$ **hasta** $\|PF_{known}\|$ **haga**
    $s_j \leftarrow PF_{known}(j)$
    $d \leftarrow \|si - sj\|_F$
    si $d < \tilde{d}_i \vee \tilde{d}_i = -1$ **entonces**
      $\tilde{d}_i \leftarrow d$
    fin si
  fin para
  $S \leftarrow S + \tilde{d}_i{}^m$
fin para
$S \leftarrow \frac{\sqrt[m]{S}}{\|PF_{true}\|}$
regresa S

2.1.10.7. Espaciamiento (*S*)

La técnica de espaciamiento (Deb, 2001), sirve como un indicador de la distribución de las soluciones PF_{Known} sobre el espacio. *S* se define de la siguiente manera:

$$S = \left[\left(\frac{1}{\|PF_{norm}\| - 1} \right) \sum_{i=1}^{\|PF_{Known}\|} \left(\bar{d} - d_i \right)^m \right]^{(1/m)}$$

En donde PF_{norm} es el valor correspondiente a los valores normalizados de PF_{Known} y d_i es definido como la distancia absoluta entre $s_i \in PF_{norm}$ y el vecino más cercano $s_j \in PF_{norm}$ para $PF_{norm} \in R^{q \times m}$, es decir:

$$d_i = min \left\{ \sum_{k=1}^{q} \left| s_{ik} - s_{jk} \right| \right\}, i \neq j$$

En Donde s_{ik} representa el componente $k-ésimo$ de la solución i.
A continuación, se presente al algoritmo con los pasos necesarios para el Espaciamiento de un Frente de Pareto Normalizado.

Algoritmo 5. Métrica de Espaciamiento *S*

Parámetros: $PF_{norm} \in \Re^{n \times m}$
$\{PF_{norm}$ Frente de Pareto conocido Normalizado$\}$
Salida: Valor S con la metrica S

$S \leftarrow 0$
para $i = 1$ **hasta** $\|PF_{norm}\|$ **haga**
  $s_i \leftarrow PF_{norm}(i)$
  $d_i \leftarrow -1$
  para $j = 1$ **hasta** $\|PF_{norm}\|$ **haga**
    si $i \neq j$ **entonces**
      $s_j \leftarrow PF_{norm}(j)$
      $d \leftarrow \sum_{k=1}^{q} |s_{i_k} - s_{j_k}|$
      si $d < d_i \vee d_i = -1$ **entonces**
        $d_i \leftarrow d$
      fin si
    fin si
  fin para
  $S \leftarrow S + d_i^m$
fin para
$S \leftarrow \sqrt[m]{\frac{S}{\|PF_{norm}-1\|}}$
regresa S

Fuente: Niño (2012)

2.1.10.8. Error de Aproximación

Dada una aproximación generada por un algoritmo, el error de aproximación describe el margen de error existente entre dicha aproximación y la solución real. La expresión que determina el error de aproximación se muestra a continuación:

$$\epsilon = \left(\left| \frac{ReGNDV - GNDV_{PF_{True}}}{GNDV_{PF_{True}}} \right| \right) \times 100\%$$

2.1.10.9. Hiper-área (*HA*)

El Híper-área (*Zitzler & Thiele, 1998)* permite obtener una medida del área de cubrimiento de PF_{Known} con respecto al espacio objetivo de un problema de optimización bi-objetivo. De manera forma el Híper-área puede expresarse como se muestra a continuación:

$$HA = \left\{ \bigcup_{i=1}^{GNDV} a_i | v_i \in PF_{Known} \right\}$$

En donde a_i es el área $i - ésima$ formada por el vecto no dominad v_i. La unión de las áreas garantizará el correcto calculo de esta métrica. La suma de áreas, tomando como base el mismo punto, por ejemplo el origen, puede estimar de manera errónea esta métrica. Cada sub-área generada por un vértica (vector no dominado) debe ser calculada teniendo en cuenta el sub-área calculada por su predecesor en el frente de Pareto (Niño, 2012).

2.2. Estado del arte

En este apartado se quiere identificar los avances en cuanto al tratamiento del problema del agente viajero, el cual, ha sido estudiado a través de diferentes técnicas, con el fin de encontrar soluciones óptimas o mejores aproximaciones, en particular se revisarán los referentes teóricos del problema del agente viajero mono-objetivo, multi-objetivo, y también los avances en la implementación de la estrategia de cluster para el tratamiento del TSP.

A continuación, se profundiza en el espacio científico-literario, relevante sobre el problema del agente viajero, teniendo en cuenta las consideraciones descritas anteriormente.

2.2.1. Problema del Agente Viajero Mono-Objetivo

El Problema del Agente Viajero mono-objetivo, como ya se ha dicho, es un problema NP-Completo (Karp, 1975). Por esta razón, el tratamiento que se le ha dado el problema ha sido variado de acuerdo al interés del investigador. En este sentido, se han utilizado tanto los métodos exactos, como los métodos de aproximación. Los métodos exactos garantizan encontrar el ciclo Hamiltoniano óptimo para el TSP, pero en un tiempo computacional exponencial. Sin embargo, debido al constante esfuerzo en el área de investigación combinatoria, se han logrado desarrollos para reducir los tiempos de procesamiento para la obtención de soluciones para el TSP. En este sentido, podemos citar a Held & Karp, (1962) y

Bellman, (1962) quienes de manera independiente propusieron un método de programación dinámica para resolver el TSP en un tiempo computacional $O(n^2 * 2^n)$. Luego de esto, Bjorklund (2010) a través del método Monte Carlo consiguió reducir este tiempo hasta $O(1.657^n)$. Otros Algoritmos exactos, que han sido usados para la solución del TSP son el Branch-and-Bound (De Klerk & Dobre, 2011), Branch-and-Cut (Hernández & Salazar, 2004) y Cutting-Plane (Levine, 2000), estos han sido probados y han generado resultados eficientes para instancias menores a 1000 ciudades, No obstante, si se tienen en cuenta un mayor número de ciudades, el tiempo computacional aumenta considerablemente (Ergun & Orlin, 2006).

Por otra parte, se tienen los algoritmos de aproximación con los cuales es posible garantizar eficiencia en el tiempo computacional, debido que por lo general aproximan las soluciones rápidamente. Estos algoritmos se dividen en métodos de construcción y métodos de mejoramiento. Los métodos de construcción generan soluciones a partir de la construcción de rutas a partir del análisis de ciudad a ciudad hasta que se construye la aproximación. El método del Vecino más Cercano (Dumitrescu & Mitchell, 2001), el Árbol de Expansión Mínimo y las Heurísiticas Christofides (Christofides, 1976) hacen parte de los métodos de construcción. Estos algoritmos se convierten en las bases para algoritmos más complejos. Los Métodos k-opt y las Heurísticas Lin-Kernighan son representaciones de los algoritmos de mejoramiento de tours. Estos han sido probados para el TSP con un alto número de ciudades y los resultados has sido bastante satisfactorios.

Otros algoritmos heurísticos han sido desarrollados en las pasadas dos décadas que han permitido mejorar la eficiencia en las aproximaciones al TSP, en este sentido podemos citar el Recocido Simulado, la Búsqueda Tabú, Algoritmos Genéticos, Colonias de Hormigas, Enjambre de Partículas, Honey Bee Mating Optimization, Redes Neuronales y algunos algoritmos híbridos Lin et al. (2016).

En los últimos años, un amplio número de estudios se han enfocado en el mejoramiento de las heurísticas tradicionales principalmente a través del desarrollo de algoritmos híbridos y generando métodos basados en mutaciones que permitan hacer el proceso iterativo más efectivo.

C.F. Tsai et al. (2004) por ejemplo, presentaron una metaheurística de aproximación llamada Algoritmo ACOMAC para resolver el Problema del Agente Viajero. Esta investigación introdujo el concepto de múltiples clanes de hormigas de algoritmos genéticos paralelos para buscar espacios de soluciones utilizando islas para evitar caer en mínimos locales y por lo tanto poder producir mínimos globales. Además, desarrollaron dos enfoques denominados el vecino más cercano múltiple y el vecino más cercano dual los cuales permiten al ACOMAC aproximar mejores soluciones para grandes instancias del TSP.

Por otra parte, Pérez (2011) propone a través del análisis sistémico y el método del Vecino Más Cercano VMC una solución al TSP. En dicha propuesta, se

establece la estrategia del sacrificio corto placista adaptativo, la cual se basa en que a veces es necesario hacer sacrificios en el corto plazo, con el ánimo de que se tenga un futuro mejor. Esta heurística se materializó, desde el punto de vista algorítmico, en el hecho de que en algún momento de la práctica del VMC, el siguiente desplazamiento no se realice a la ciudad inmediatamente cercana, disponible, sino que se renuncie a ella para trasladarse hacia la segunda más cercana disponible y, a partir de este cambio, se continúe con la tradicional regla VMC. Esta estrategia propuesta, atendiendo a una de las tendencias arrojadas por la revisión de literatura, llevó a complementar el sacrificio corto placista adaptativo con una búsqueda local 2opt. Los resultados fueron comparados con otras los obtenidos por otras heurísticas, y se encontraron mejores soluciones que otras heurísticas reconocidas por la literatura.

De otro lado, Cockbaine y Silva (2013) presentan una mejora para los algoritmos heurísticos basado análisis de los resultados a través de indicadores. La construcción de indicadores permite que la retroalimentación con ello se logran mejores soluciones. Para concretar esta estrategia, fue necesario fusionar especificaciones de la heurística, seudocódigo y código fuente, además de añadir en el código fuente llamados a una función generadora de datos, imprescindible para concretar la formulación de los indicadores. La estrategia fue aplicada a un algoritmo heurístico, obteniéndose una versión mejorada del mismo, que logró mejores soluciones en promedio que su primera versión.

Trabajos más actuales presentan combinación de heurísticas con estrategias que mejoren el proceso de búsqueda. En este sentido podemos citar a Lin et al. (2016) cuyo aporte se basa en el desarrollo de un algoritmo metaheurístico híbrido adaptativo que combina algoritmos de recocido simulado y búsqueda tabú con una estructura de vecindario dinámico para resolver el TSP. Los resultados experimentales demostraron que el algoritmo propuesto puede obtener soluciones satisfactorias dentro de un tiempo razonable. Además, el algoritmo propuesto puede superar las desventajas individuales del recocido simulado y la búsqueda de tabú. El nuevo algoritmo híbrido proporciona una tasa de disminución rápida y un proceso de convergencia claro, y no depende de soluciones iniciales. Además, en comparación con el clásico vecindario 2-opt, el vecindario dinámico puede reducir significativamente el tiempo computacional, disminuyendo el número de cálculos y, simultáneamente, mejorando la calidad de la solución.

Ozden et al. (2017) por su parte, proporciona un método práctico y eficaz para resolver computacionalmente instancias grandes del TSP. En esta investigación, se describen como técnicas de computación paralela, las cuales pueden disminuir significativamente el tiempo computacional general y aumentar la utilización de la CPU para resolver grandes lotes de TSP utilizando métodos de clase paralelos simples y efectivos en C #. Es importante resaltar en primer lugar, que puede utilizar diferentes soluciones de TSP para probar un método exacto y una heurística. Asimismo, puede manejar TSP de diferentes tamaños eficientemente con una simple regla de procesamiento. Este problema surgió del diseño de

transporte redes, redes de distribución y almacenes. De esta manera, los autores describen y comparan implementaciones de Solver en serie, paralelos y distribuidos, teniendo en cuenta grandes lotes de problemas del TSP usando la Heurística de Lin-Kernighan (LKH) y el Solucionador de TSP exacto Concorde.

Asimismo, con el fin de mejorar la eficiencia y la eficacia de las soluciones, Prasad y Mukherjee (2016) presentan el algoritmo de búsqueda de organismos simbióticos (SOS), este es un nuevo algoritmo de búsqueda metaheurística eficaz, que recientemente ha registrado una aplicación más amplia en la solución de complejos problemas de optimización. SOS imita las estrategias de relación simbiótica adoptadas por los organismos en el ecosistema para la supervivencia. En este trabajo, se presenta un estudio sobre la aplicación de SOS con Recocido Simulado para resolver el problema del Agente Viajero. La intención del método híbrido propuesto, es evaluar el comportamiento de convergencia y la escalabilidad de la búsqueda del organismo simbiótico con recocido simulado para resolver el TSP tanto para instancias pequeñas como para instancias de gran escala. Las prestaciones de este algoritmo se evaluaron en diferentes conjuntos de patrones de TSP obtenidos de TSPLIB (una biblioteca que contiene muestras de instancias de TSP). Los resultados del análisis empírico muestran que la calidad de los resultados finales, así como la tasa de convergencia del nuevo algoritmo, en algunos casos produjeron soluciones superiores que los mejores resultados conocidos de TSP.

Como vemos, el desarrollo en cuanto a metodologías, técnicas, y heurísticas para darle solución al Problema del Agente Viajero, es variado y se ha mantenido en constante desarrollo, con el fin de generar mayor eficiencia y efectividad en las soluciones. Cada investigación nueva desafía las mejores soluciones presentadas en la literatura, y en algunos casos las supera usando Metaheurísticas más efectivas, sin embargo, debido a que no se tiene conocimiento del óptimo para todas las instancias del TSP, aún es pertinente que la comunidad científica siga trabajando para optimizar este problema combinatorio.

2.2.2. Problema del Agente Viajero Multi-Objetivo

La optimización combinatoria encuentra aplicaciones en muchas áreas como programación de la producción, programación de proyectos, programación de personal, ruteo de vehículos, planeación de inversiones, entre otros. En la vida real, las soluciones a estos problemas combinatorios tienen en cuenta muchos diferentes puntos de vista, correspondientes a diferentes objetivos que se quieren lograr. Estos objetivos, en ocasiones entran en conflicto entre sí. Por esta razón, la optimización multi-objetivo cobra sentido, ya que su propósito es encontrar una solución que pueda ser la mejor posible para cada uno de los objetivos que se quiere alcanzar. Usualmente, no hay una solución individual que optimice todos los objetivos simultáneamente, la selección de una solución dependerá de las

preferencias del analista. Esta decisión la debe tomar teniendo en cuenta un conjunto de soluciones eficientes. Por lo tanto, los métodos de optimización multi-objetivo deben reducir el espacio de búsqueda al conjunto de soluciones eficientes.

Debido a la complejidad computacional de los problemas combinatorios multi-objetivos, se necesitan metaheurísticas, tales como Algoritmos Genéticos, Recosido Simulado o Búsqueda Tabú, las cuales pueden ser bastante útiles en la generación de soluciones eficientes.

En este sentido Jaszkiewicz, A. (2002), propuso un algoritmo de búsqueda local genética para la optimización combinatoria multi-objetivo. El objetivo del algoritmo es aproximar en un corto tiempo un conjunto soluciones eficientes que permitan al analista tomar una decisión adecuada. En cada iteración el algoritmo dibuja al azar una función de utilidad y construye una población temporal compuesta por un número de mejores soluciones entre las soluciones generadas hasta ese momento. Luego un par, de soluciones seleccionadas de manera aleatoria de la población. Finalmente, se aplica un método de búsqueda local a cada descendiente. Los resultados encontrados con esta metaheurística superan a otros métodos de búsqueda local genética presentes en la literatura y un algoritmo genético basado en el ranking de clasificación de Pareto para el TSP.

Mora et al. (2013) en su investigación “Pareto-based multi-colony multi-objective ant colony optimization algorithms: an island model proposal”, presenta un estudio sobre diferentes esquemas de distribución de grano grueso que tratan con Algoritmos de Optimización Multi-Objetivos de Colonia de Hormigas. Dos de ellos son una variación de estructuras de múltiples colonias independientes, que tienen respectivamente un número fijo de hormigas en cada subconjunto o que distribuyen la cantidad total de hormigas en pequeñas sub-colonias.

Recientemente, Ariyasingha y Fernando (2016) presentaron una modificación del algoritmo de Colonias de Hormigas de Pareto para resolver dos problemas de optimización combinatoria: el Problema del Agente Viajero (TSP); Y el problema de programación de taller (JSSP). El método que utilizan está basado en el método de peso aleatorio lo cual es tomado como una mejora. El método propuesto logró un mejor desempeño que el algoritmo PSACO original para ambos problemas de optimización combinatoria y como puede verse en la siguiente figura, obtuvo frentes Pareto-óptimos bien distribuidos.

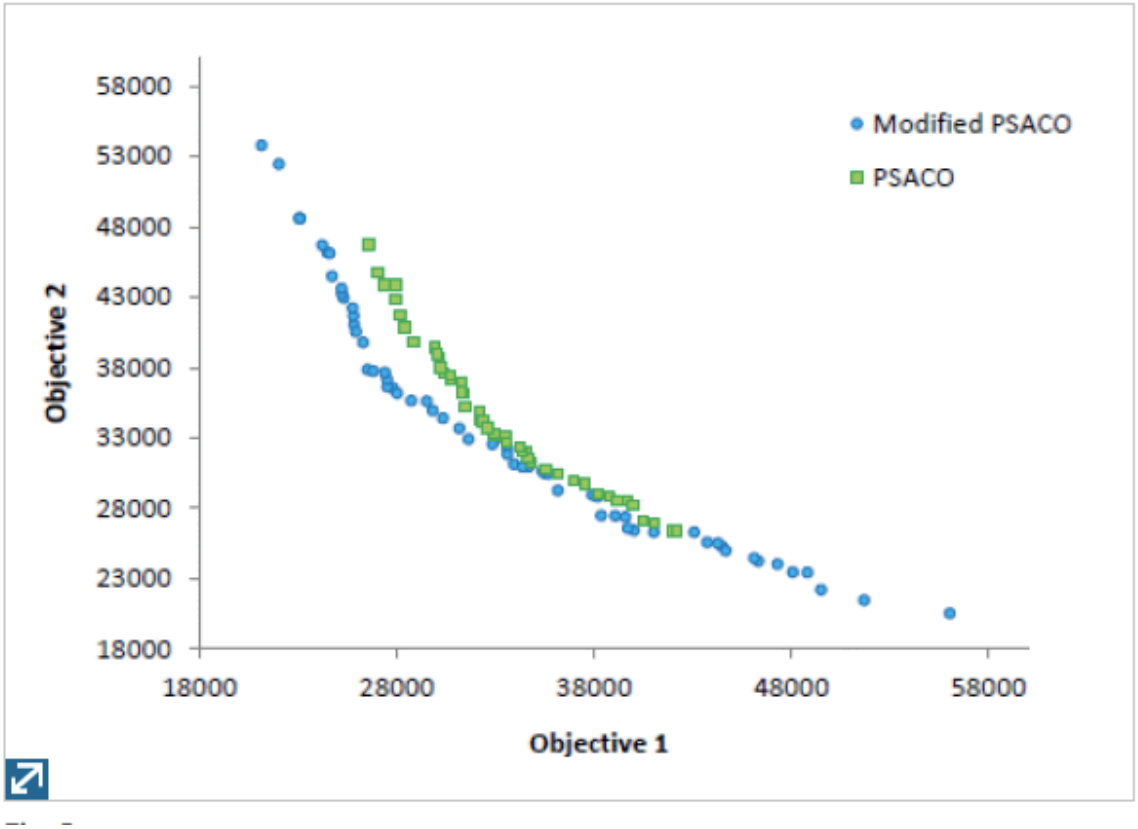

Figura 7. Soluciones no dominadas generadas por cada algoritmo para las instancias TSP Kroac50.

Fuente: Ariyasingha y Fernando (2016)

2.2.3. Problema del Agente Viajero con Clusters

El primer acercamiento al tratamiento del problema del agente viajero mediante clusters, lo hizo Chisman, (1975), el cual presentó el CTSP (Clustered traveling salesman problem), como una variante del problema clásico del agente viajero. En este, un conjunto de vértices (exceptuando el vértice de salida o deposito) son divididos en algunos clusters pre-especificados. El objetivo es obtener un ciclo Hamiltoniano de menor costo visitando cada una de las ciudades y visitando cada uno de los clusters en un orden predefinido.

Estudios posteriores presentaron aplicaciones del CTSP en ruteo automatizado de bodegas (Chisman, 1975), planeación de la producción (Lokin, 1978), y problemas de ruteo con niveles de prioridad asociado a los clientes. Algoritmos genéticos también han sido utilizados para solucionar el CTSP (Potvin & Guertin, 1996).

Laporte et al. (1997), hicieron un aporte a la solución del CTSP, introduciendo una heurística de búsqueda tabú para resolver este problema. Este algoritmo reinicia periódicamente su búsqueda combinando dos soluciones de élite para formar una nueva solución inicial (de una manera que recuerda a los algoritmos genéticos).

Bao, X., & Liu, Z. (2012), nos presentan un método mejorado para la solución del problema del agente viajero agrupado (o CTSP: Clutered Travelin Salesman Problem). El objetivo es calcular un ciclo más corto, de modo que todos los vértices son visitados y los vértices de cada cluster se visitan consecutivamente, suponiendo que no se especifican vértices iniciales y finales de ningún cluster. El método permite determinar el orden de los clusters, así como el orden de los vértices en cada cluster. En esta implementación, el algoritmo genera dos rutas diferentes y escoge la mejor.

Mestria et al. (2013) propuso heurísticas basadas en heurística GRASP para resolver el CTSP con distancias simétricas y euclidianas, incorporando estrategias de re-direccionamiento de rutas. Este estudio, ha ido evolucionando, de tal manera que Mestria et al. (2016) ha implementado un nuevo algoritmo hibrido basado en la metaheurísticas GRASP, Iterated Local Search y Variable Neibhorhood Descent. Cabe resaltar, que estos estudios se han enfocado en instancias de 2000 ciudades.

Por otra parte, este problema también ha sido tratado a través de un algoritmo genético hibrido, presentado por Ahmed, Z. H. (2014), el cual utilizó crossover constructivo secuencial, búsqueda de 2-opt, una búsqueda local, y un método de inmigración para obtener una solución heurística para el TSP con clusters ordenados u OCTSP (por sus siglas en inglés). El problema ordenado del agente viajero agrupado, es una variación del problema usual del TSP en el que un conjunto de vértices (excepto el vértice inicial) de la red se divide en clusters pre-especificados. El objetivo es encontrar el ciclo hamiltoniano de menor costo en la que los vértices de cualquier cluster se visitan contiguamente y los clústeres se visitan en el orden pre-especificado.

Actualmente, se han hecho nuevos acercamientos al tratamiento del problema del agente viajero a través de clusters, así por ejemplo, Li, X., & Liu, J. B. (2017), nos presentan un mecanismo de estudio para el TSP, que es un algoritmo de agrupación llamado Vertex Coloring Clustering (VCC), que utiliza el pensamiento de coloración de vértices para clasificar todas las ciudades en algunas clases y permite que la heurística de colonias de hormiga actúe en cada clase para obtener rutas TSP locales y luego unirlas como una ruta entera.

De otra mano, podemos evidenciar en la aplicación de Ferreira et al. (2017), el uso de una metodología de dos fases a un problema de enrutamiento, en la siguiente forma: 1) definir clusters de puntos de demanda a ser servidos, como un problema de ubicación de instalaciones (FLP); y 2) la definición de las rutas que se desarrollarán dentro de cada cluster, como un TSP con distancias asimétricas. Para una comprensión más clara, la metodología se aplicó a una empresa de

transporte rápido, haciendo uso de las metaheurísticas Simulated Annealing (SA), Tabu Search (TS) y un Algoritmo Híbrido (HA). Se evaluaron dos escenarios, con variación en el número de clusters.

Las metodologías son variadas con respecto al tratamiento del TSP mediante clusters. Esto garantiza, que los avances que realicemos desde el enfoque que queremos proponer tenga un punto de comparación, a su vez, podemos notar que no hay acercamientos al problema multi-objtetivo del TSP, lo cual, viene a ser una oportunidad para que esta investigación despierte interés en la comunidad académica y científica.

3. DISEÑO DE LA METAHEURÍSTICA MULTI-OBJETIVO

En esta investigación hemos planteado una metaheurística de optimización combinatoria para la aproximación de soluciones al Problema del Agente Viajero Multi-Objetivo. Esta metaheurística se desarrolló teniendo en cuenta una estrategia de cluster de ciudades. El objetivo de esto, es lograr tener mayor control sobre los procesos de búsqueda de aproximaciones. Tal como se muestra en la siguiente figura, la idea gráfica de la estrategia de cluster es dividir el espacio de soluciones para luego implementar el método del Vecino más Cercano dentro de cada uno de esos cluster.

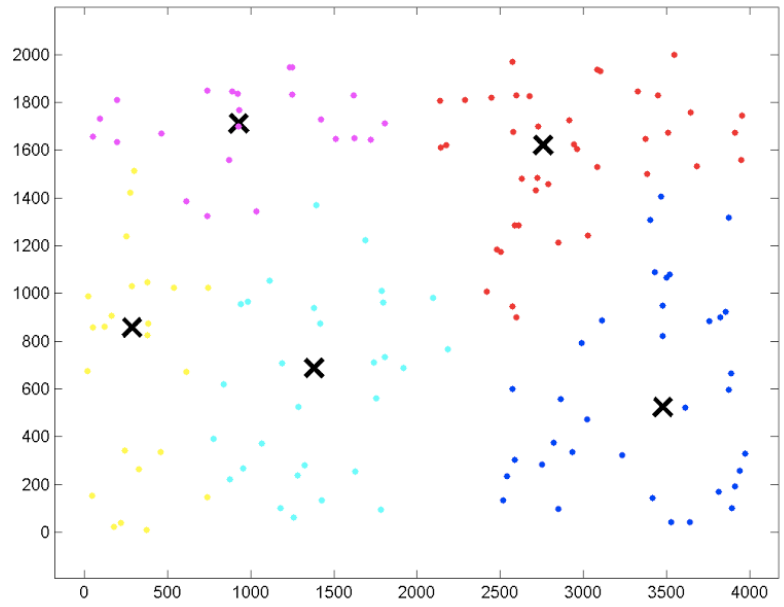

Figura 8. Representación gráfica del cluster de ciudades

El cluster de ciudades se genera a través del método K-Medoids, el cual como se explicó en la sección 2.1 divide un conjunto de mediciones u observaciones en k subconjuntos o clusters, de modo que los subconjuntos minimicen la suma de las distancias entre una medida y un centro de la medida del cluster. En el algoritmo k-medoids, el centro del subconjunto es un miembro del subconjunto, llamado medoid.

3.1. Datos de entrada

En el problema del agente viajero multi-objetivo, se pueden presentar como parámetros de entrada datos con unidades de distancia, costo, capacidad, tiempo entre otros, los cuales deben ser normalizados para construir las matrices de peso de cada función objetivo. No obstante, para esta investigación no tendremos que normalizar unidades, debido a que las matrices provienen de instancias que tienen la misma unidad de medida.

Para efectos del proceso de optimización, el algoritmo genera una matriz que es la combinación lineal convexa de las matrices iniciales. Esto es válido para optimización multi-objetivo a través de la técnica de suma con pesos.

$$F(X) = \alpha f_1(x) + (1 - \alpha) f_2(x)$$

Cabe resaltar que, para efectos de comparar la calidad de los resultados, se trabajó con instancias las presentadas por las librerías de soluciones TSPLIB, específicamente se utilizaron las instancias KRO para 100 ciudades. Cabe notar que, TSPLIB presenta instancias mono-objetivo. Por ende, para generar instancias bi-objetivo, tomamos los puntos de dos instancias y generamos una matriz de peso para cada instancia, las cuales serán nuestras matrices de las funciones objetivos 1 y 2. Asimismo, Las matrices de pesos, se obtienen determinando la distancia Euclidiana entre cada uno de los puntos presentados en la instancia.

3.2. Algoritmo propuesto

A continuación, se presenta la lógica principal que permite el desarrollo de la Metaheurística. En esta se describe en primer lugar los datos de entrada, que se refiere a las matrices de pesos de las funciones objetivos 1 y 2. El resultado, es un vector de soluciones para cada valor de α, es decir una pareja ordenada, que representa el mejor valor para f_1y f_2 , el cuál, hará parte del frente de Pareto. El número de particiones k, que va de 2 hasta 8, va aumentando una unidad en cada iteración, con el fin de mejorar el proceso de optimización de acuerdo a dichas

particiones. Es también importante hablar de la variable α, la cual incrementa de 0 hasta 1. El valor del incremento de α, dependerá del número de soluciones que se desee generar. De esta manera, para un incremento de 0.01, la metaheurística generará 100 soluciones. Es importante resaltar que, para cada α, se genera un ciclo de optimización. Lo anterior, va a permitir que se genere un Frente Pareto uniformemente distribuido, porque el valor de α direccionará las soluciones.

Algoritmo 6. Principal

Algoritmo: Main

Datos: MObjetivo1, MObjetivo2
Resultado: Valores óptimos Multi-objetivo
Inicialización;
Se define el número de K particiones.
$F_{óptimo} = \infty$.
para α *=0 hasta 1, con incrementos de* δ **hacer**
  M=$MObjetivo_1 * \alpha + MObjetivo_2 * (1 - \alpha)$
  para *k=2 hasta el número de particiones K* **hacer**
    Creamos los Clusters;
    para *Número de perturbaciones* **hacer**
      Se Aplica búsqueda Tabú para refinar los recorridos obtenidos al aplicar vecino más cercano (VMC) a cada cluster en su respectiva ciudad inicial.
      $F_{actual} = \infty$.
      para *i=1:k* **hacer**
        Aplicamos VMC a centroides para tener una secuencia de recorrido.
        Variamos el Cluster k_i de Inicio.
        Se unen los clusters.
        Se calcula F_i
        si $F_i < F_{actual}$ **entonces**
          Actualizamos valor actual. Actualizamos Ruta actual.
        fin
      fin
      si $F_{actual} < F_{óptimo}$ **entonces**
        Actualizamos valor óptimo. Actualizamos Ruta .
      fin
      Generamos aleatoriamente t nuevas ciudades de origen.
    fin
    Guardamos solución para valor de α en un vector
    Repetimos la ruta óptima en MObjetivo1 y MObjetivo2 para obtener las coordenadas (x,y) del Frente de Pareto.
  fin
fin

El algoritmo propuesto tiene tres componentes principales; el primero es la generación de los *k* clusters a través de K-medoid a partir de la matriz combinación lineal M.

Algoritmo 7. Creación de Cluster

Algoritmo: Creación de Clusters

Datos: Matriz M, Numero K de particiones
Resultado: Índices de Clusters,Ciudades inicio, Índices Centroides
Se aplica K-medoids a M para obtener los K clusters.
para *k=1 hasta el número de particiones* **hacer**
    Se obtienen los índices del cluster k
    Se genera una ciudad de inicio de manera aleatoria para el cluster k
fin

Dentro de cada uno de estos cluster se genera una solución inicial utilizando la heurística del vecino más cercano, pero con la condición que no cierre la ruta en ningún caso. Esta primera fase, claramente no va a generar un óptimo, pero el objetivo es aportar una solución que puedas refinarse a través de un algoritmo de búsqueda Tabú. Aplicar el vecino más cercano a nivel de clusters, va a reducir el efecto de miopía que se generaría al aplicarlo al conjunto total de ciudades.

El segundo componente, se genera una ruta completa uniendo cada uno de los cluster que están inicialmente abiertos. Para cumplir este objetivo, se aplica la heurística del Vecino más Cercano teniendo en cuenta los Medoids (o puntos de referencia para generar los clusters) definidos previamente. Esto genera una secuencia para cada uno de los cluster, y simultáneamente una ruta total inicial.

Algoritmo 8. Generar unión de clusters

Algoritmo: Unir Caminos de los Clusters

Datos: Rutas de clusters,Secuencia,Matriz M, Numero K de Clusters,Ciudades Origen-Fin,Permutar

Resultado: Ruta óptima, Costo óptimo

para *k=1 hasta el número de particiones* **hacer**

  Obtener Rutak para el cluster k.

  si *Permutar=1* **entonces**

    Invertir Rutak

  fin

  si $k < K$ **entonces**

    Aplicamos VMC a la ciudad origen del Cluser k, con respecto a las ciudades libres en el siguiente de la secuencia.

    Indicamos la necesidad de permutar

  fin

  Se añade la ruta.

fin

Se evalúa la ruta.

El tercer componente, es la implementación de la búsqueda Tabú como estrategia de búsqueda local con el fin de reducir la cantidad de soluciones factibles a considerar, concentrándose específicamente en las mejores soluciones. La búsqueda Tabú se utiliza en dos momentos del proceso de optimización. En primer lugar, se utiliza para permutar el orden en que son visitadas algunas de las ciudades dentro de cada uno de los clusters. En segundo lugar, se utiliza para permutar el orden de secuencia de los cluster.

Algoritmo 9. Búsqueda Tabú entre clusters

Algoritmo: búsqueda Tabú dentro de los Clusters

Datos: Índices de Cluster, Número de K particiones, Matriz M,Ciudades iniciales , Número de Perturbaciones

Resultado: Rutas óptimas de clusters, Ciudades desconectadas

para *k=1 hasta el número de particiones* **hacer**

  Obtener índices del cluster k.

  Obtener matriz M local.

  Aplicar VMC a Cluster k.

  si *Cantidad de ciudades en cluster k >1* **entonces**

    Aplicar búsqueda Tabú a ruta obtenida

  en otro caso

    Guardar ruta

  fin

  Guardar ciudades de inicio y fin.

fin

4. VALIDACIÓN DE LA METAHEURÍSTICA

Para la etapa de experimentación, se probó la metaheurística propuesta con instancias del Problema del Agente Viajero, el cual como ya se ha dicho, está catalogado como NP completo, de él se derivan gran cantidad de sub-problemas concernientes a diversos campos de la ingeniería.

Las instancias de los problemas fueron tomadas de la librería de soluciones para el Problema del Agente Viajero, TSPLIB. Esta librería contiene más de 200 instancias, adicionalmente alberga otras 500 instancias de problemas conocidos tales como el Ruteo de Vehículos (problema derivado del TSP). TSPLIB es administrada por el Grupo de Investigación de Optimización Discreta de la Universidad de Heidelberg en Alemania (135 a nivel mundial). Más de 5000 artículos científicos utilizan estas instancias para probar las técnicas propuestas y a su vez, contrastar los resultados obtenidos con otros autores (Niño, 2010). Lo anterior, podemos asegurar que TSPLIB es una buena opción para probar y contrastar los resultados obtenidos por medio de la Metaheurística propuesta con otras técnicas de talla mundial. Las instancias utilizadas de esta librería se presentan en la siguiente tabla.

Tabla 1. Instancias de la TSPLIB usadas las pruedas de la Metaheurística

Nombre de la Instancia	Número de Ciudades	Número de Objetivos
KROAB100	100	2
KROAC100	100	2
KROBC100	100	2
KROCD100	100	2

Los resultados obtenidos se contrastan con otras técnicas utilizando las métricas explicadas en el capítulo 2.

4.1. Especificaciones de la Metaheurística

Para llevar a cabo la prueba es necesario definir los parámetros de ejecución del algoritmo.

La cantidad de iteraciones externas (Q), será de 100. Esto es el número de permutaciones realizadas por la búsqueda Tabú con respecto a los cluster.

La cantidad de iteraciones internas (N), será de 100. Esto es el número de permutaciones realizadas por la búsqueda Tabú dentro de cada uno de los cluster. El valor α que va de 0 a 1, aumentará en 0.1 en cada iteración.

El número de particiones *k* que va de 2 a 8, aumenta una unidad en cada iteración.

La función objetivo F(x), es la combinación lineal de los dos objetivos a optimizar.

$$F(X) = \alpha f_1(x) + (1 - \alpha) f_2(x)$$

4.2. Comparación de resultados de la Metaheurística propuesta.

En esta etapa de la prueba, se contrastaron los resultados de la metaheurística propuesta contra los resultados obtenidos por algoritmos reconocidos por la comunidad científica y académica en problemas combinatorios multi-objetivo, como son: BCA, BCM, CA, MACS, MOAQ, MONA, NSGA-II, PACO, SPEA2 y USBC.

García-Martínez et al. (2007), implementador de los algoritmos BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, corrió 10 veces cada uno de los algoritmos y al final recolectó las soluciones no dominadas de todas las corridas. Con el fin de hacer una comparación eficiente, solo se analizarán los frentes de Pareto con las soluciones no dominadas del conjunto obtenido de la unión de cada uno de los frentes de Pareto de cada algoritmo. Asimismo, se tendrán en cuenta las instancias, KROAB100, KROAD100 y KROBC100.

Los algoritmos contra los cuales se compara la heurística propuesta, son los presentados por García-Martínez et al. (2007).

En la tabla a continuación, se muestra el nombre de cada uno de los algoritmos y su respectiva abreviación:

Tabla 2. Lista de Algoritmos y sus abreviaciones.

NOMBRE DEL ALGORITMO	ABREVIACIÓN
Tabu Search using K-Medoid and Nearest Neighbor	TS-KNN
Bi-Criterion Ant	BCA
Bi-criterion MC	BCM
COMPETAnts	CA
Multiple Ant Colony System	MACS
Multiobjetive Ant-Q Algorithm	MOAQ
Multiobjetive Network Optimization based on Ant Colony	MONA - MONACO
Non-dominated Sorting Genetic Algorithm II	NSGA2
Population Based Ant Colony	PACO
Strength Pareto Evolutionary Algorithm 2	SPEA2
Unsort Bicriterion Algorithm	USBC

A continuación, se presentan los frentes de Pareto obtenidos con las soluciones no dominadas de cada uno de los algoritmos, para las instancias KROAB100, KROAD100, KROBC100.

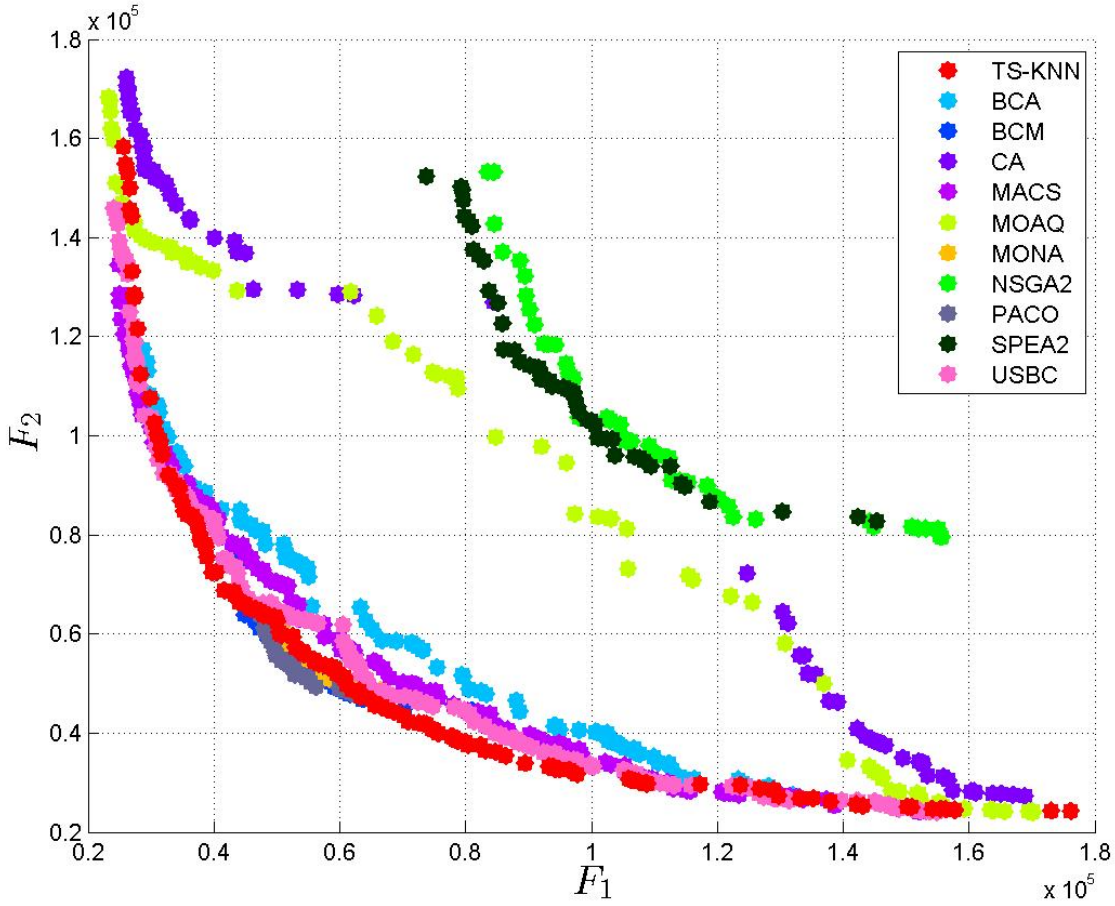

Figura 9. Frentes de Pareto obtenidos por los Algoritmos TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, en la instancia KROAB100.

Como podemos darnos cuenta en la gráfica anterior, el algoritmo TS-KNN genera soluciones no dominadas en regiones del frente Pareto ideal donde los otros algoritmos no generan. Asimismo, se puede notar que para ciertos intervalos donde las soluciones de TS-KNN no domina, se mantienen en una región cercana al mejor frente mostrado por otro algorimo.

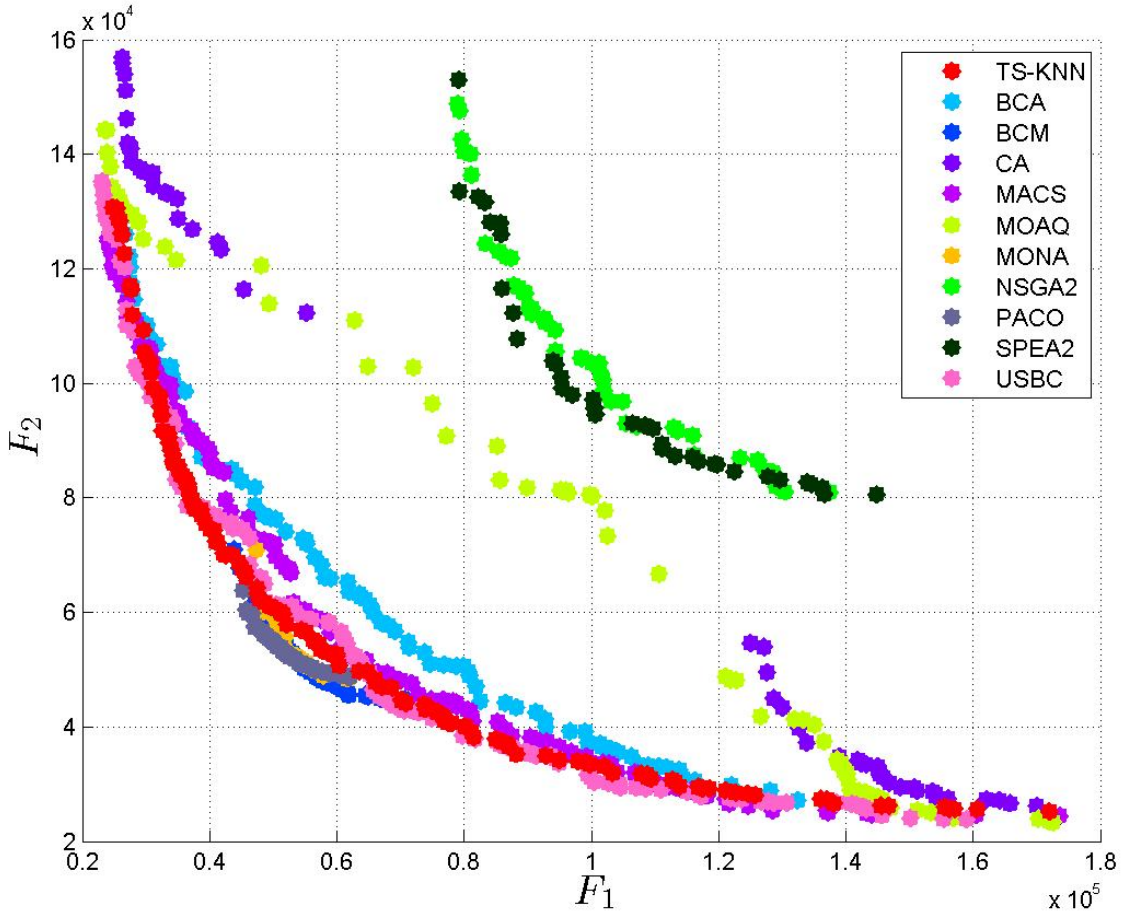

Figura 10. Frentes de Pareto obtenidos por los Algoritmos TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, en la instancia KROAD100.

Con respecto a la instancia KROAD100, el comportamiento es muy similar al anterior. El frente de Pareto de TS-KNN, se mantiene dentro de las soluciones dominantes, sin embargo, BCM, PACO, y MONA, generan mejores soluciones en el intervalo [40000, 60000] del objetivo 1, pero como se puede notar, es una región pequeña, porque TS-KNN sigue presentando mejores soluciones en direcciones BCM, PACO, y MONA no logran.

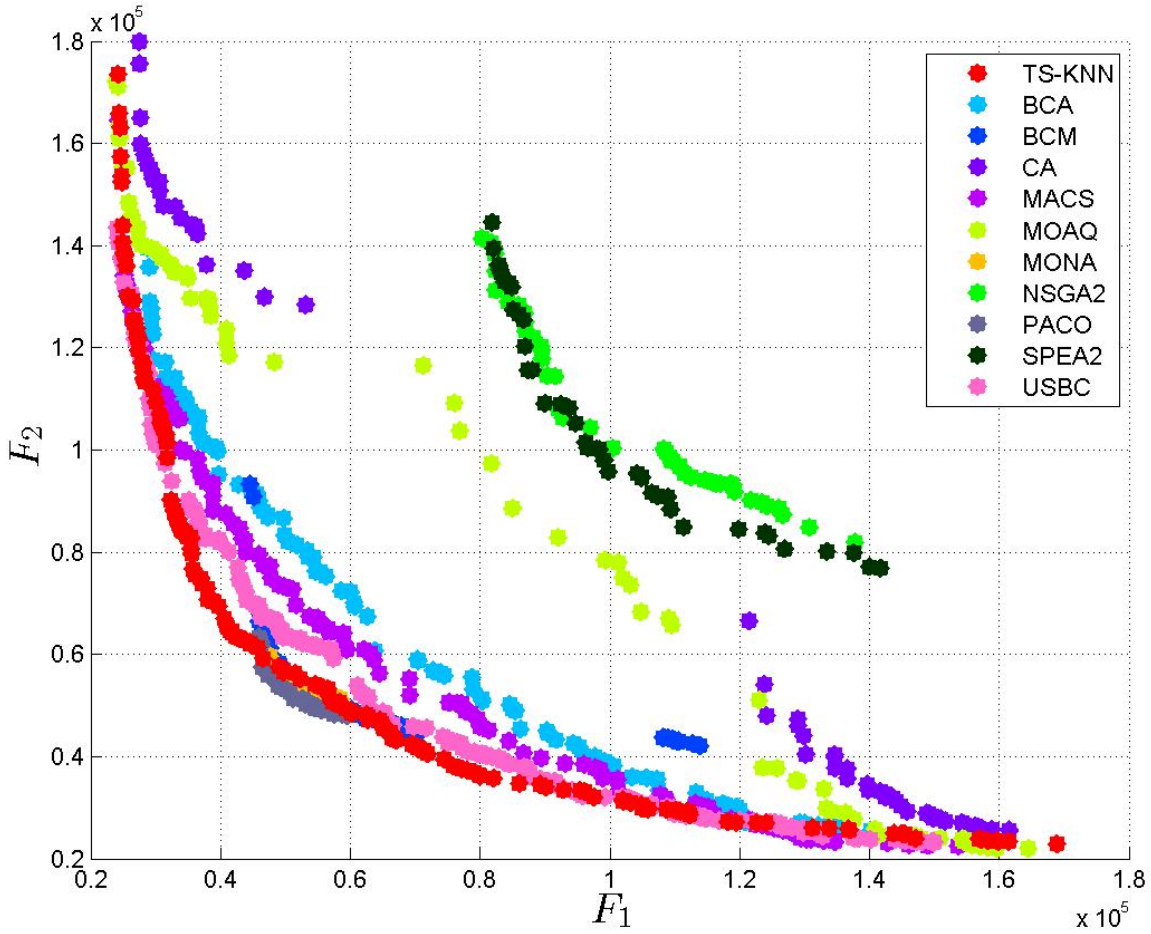

Figura 11. Frentes de Pareto obtenidos por los Algoritmos TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, en la instancia KROBC100.

Para la instancia KROBC100, la metaheurística TS-KNN nuevamente se mantiene en el área de soluciones dominantes, y claramente genera mejores resultados para los intervalos [20000, 40000] y [60000, 80000].

Con el fin de hace un análisis más preciso a la calidad de las soluciones del algoritmo TS-KNN comparado con los algoritmos citados previamente, a continuación, se presentan los gráficos de caja y bigotes, con respecto a las métricas Distancia Generacional (*GD*), Distancia Generacional Inversa (*IGD*) y Espaciamiento (en inglés Spacing).

De manera general, la métrica de distancia generacional nos va a permitir verificar, en promedio, la distancia de cada uno de los frentes de los algoritmos (PF_{known}), contra e Frente de Pareto Ideal (PF_{true}).

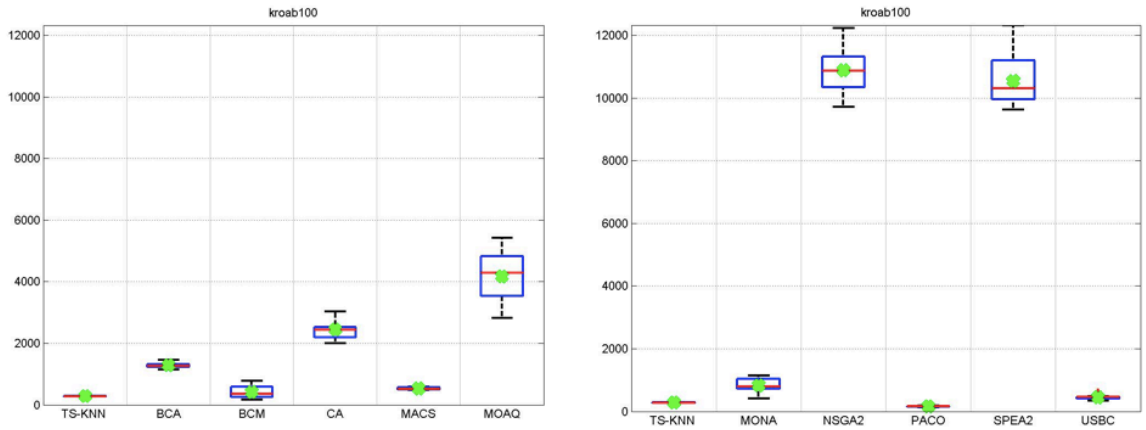

Figura 12. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional con la instancia KROAB100.

Para la instancia KROAB100, podemos notar que la metaheurística TS-KNN, presenta una media muy cercana a cero y que está por debajo de los algoritmos BCA, BCM, CA, MOAQ, MACS, MOAQ, MONA, NSGA2, SPEA2 y USBC con respecto a la métrica Distancia Generacional. En el caso del algoritmo PACO, existe una una pequeña diferencia, la cual podría mejorar con mejorando los parametros del experimento pero, es necesario resaltar que TS-KNN genera soluciones no dominadas en regiones donde PACO no genera. Por otro lado, la variabilidad que presentan las soluciones es pequeña, y esto permite que las soluciones sean en la mayoría de casos, muy cercanas al valor de la media.

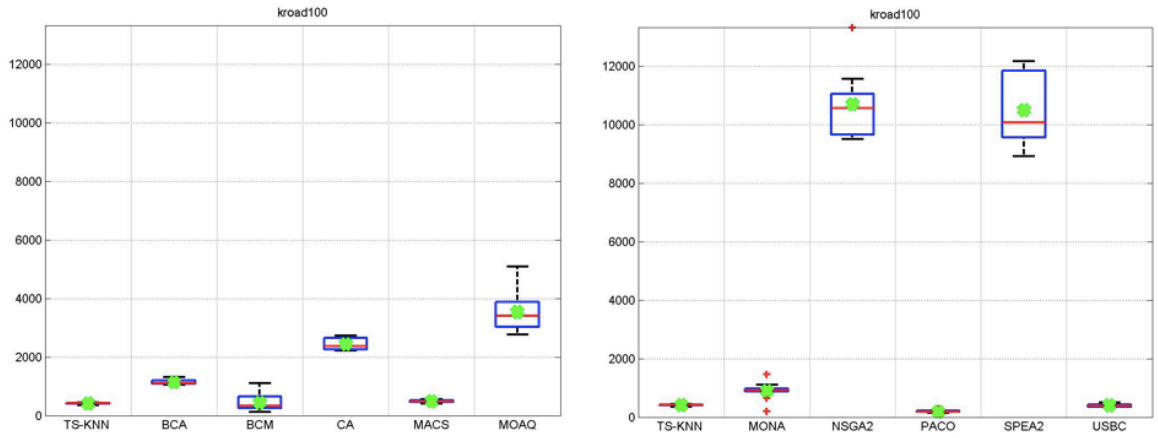

Figura 13. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional con la instancia KROAD100.

Para KROAD100, TS-KNN mantiene una media cercana a cero, y que solo es superado por los algoritmos BCM, y PACO, pero de acuerdo al análisis del frente de Pareto, TS-KNN genera soluciones dominantes en más regiones que BCM y PACO. Con respecto a la variabilidad de las soluciones, es bastante baja. Esto garantiza que las soluciones sean en la mayoría de casos cercana a la media.

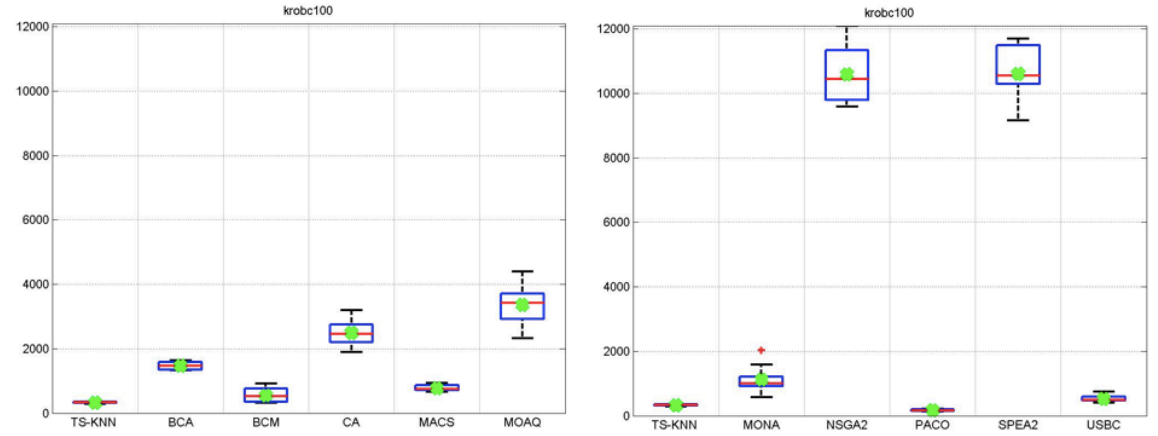

Figura 14. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional con la instancia KROBC100.

Para la instancia KROBC100 (Figura 14.), el algoritmo TS-KNN mostró mayor calidad respecto a la instancia anterior, debido a que su media está por debajo de los demás algoritmos, a excepción del PACO, el cual, para este experimento logró mostrar buen resultado, pero en los frentes de Pareto, vemos que PACO solo domina en una pequeña región. Por otra parte, TS-KNN sigue teniendo poca variabilidad, lo cual lo evidencia que es un algoritmo cuyas soluciones no se alejan de la media.

Seguidamente, se presentan los cuadros de caja y bigotes para la métrica Distancia Generacional Inversa. De manera general, se sabe que esta métrica permite conocer la mínima distancia entre el Frente de Pareto Ideal (PF_{true}) y el Frente de Pareto conocido (PF_{known}).

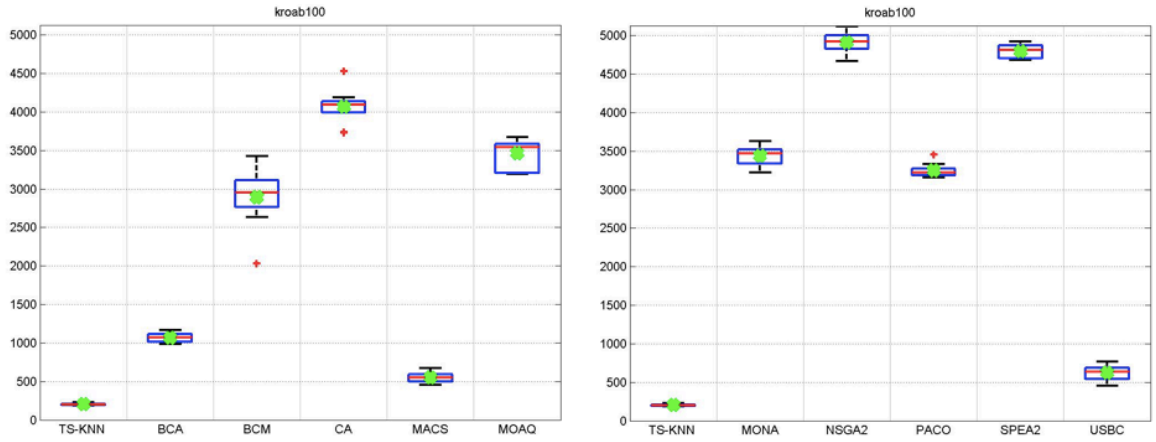

Figura 15. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional Inversa con la instancia KROAB100.

Como puede verse, para la instancia KROAB100, la metaheurística TS-KNN presenta una media bastante buena, debido a que es la más cercana cero. Asimismo, presenta menor varianza, lo cual nos dice que el total de sus datos están muy cercanos al valor de la media.

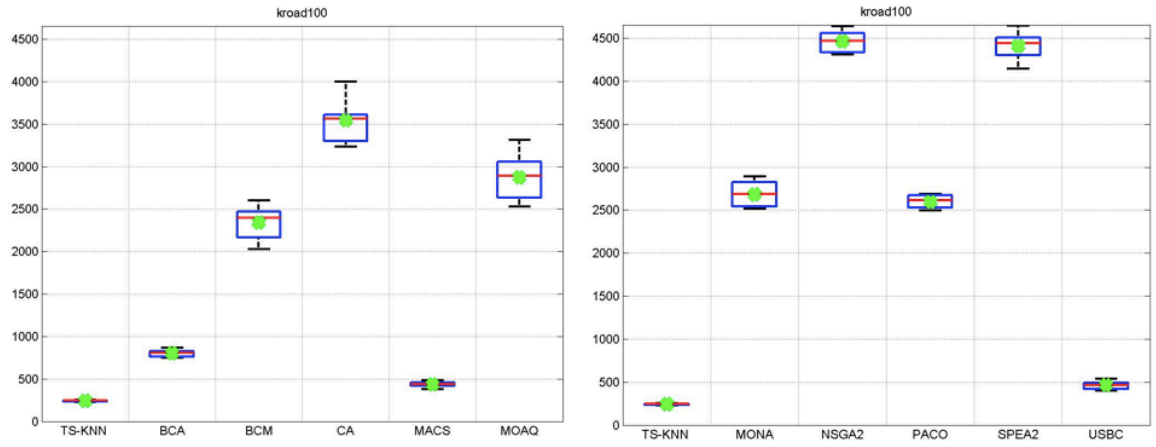

Figura 16. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional Inversa con la instancia KROAD100.

El comportamiento en la instancia KROAD100 es igual al descrito en la instancia anterior. TS-KNN, sigue siendo el método con la mejor media comparado con BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC.

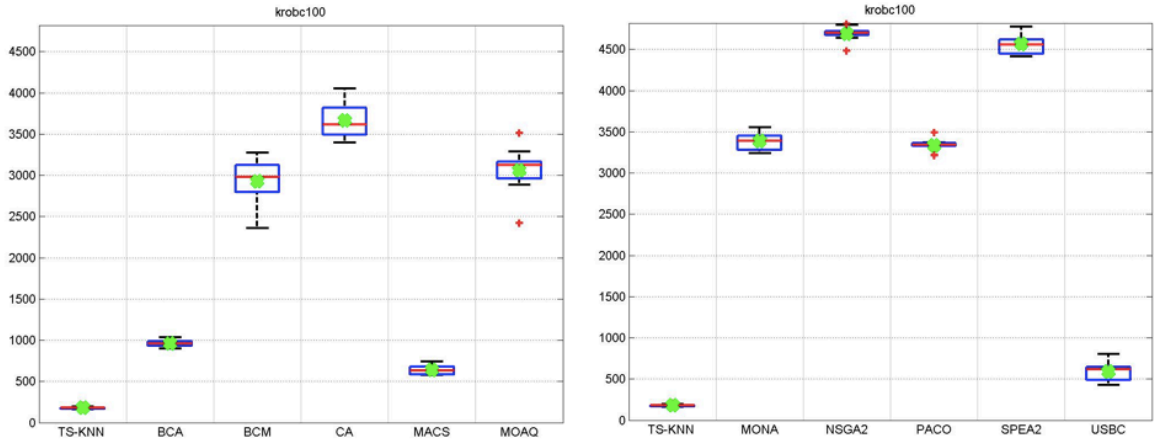

Figura 17. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Distancia Generacional Inversa con la instancia KROBC100.

Por último, se puede evidenciar que tal como lo fue en las instancias anteriores, TS-KNN, presentó la mejor media para la métrica Distancia Generacional Inversa.

Con el fin de dar un mejor análisis, a continuación analiaremos la métrica Espaciamiento para cada uno de los algorimos citados anteriormente, y los compararemos con TS-KNN, para las instacias KRO descritas previamente.

Si analizamos la Figura 21, 22 y 23, notaremos para esta instancia TS-KNN presenta resultados bastante buenos ya que su media y su varianza son bajas, y esto significa que nuestras soluciones están bien distribuidas en el frente de Pareto.

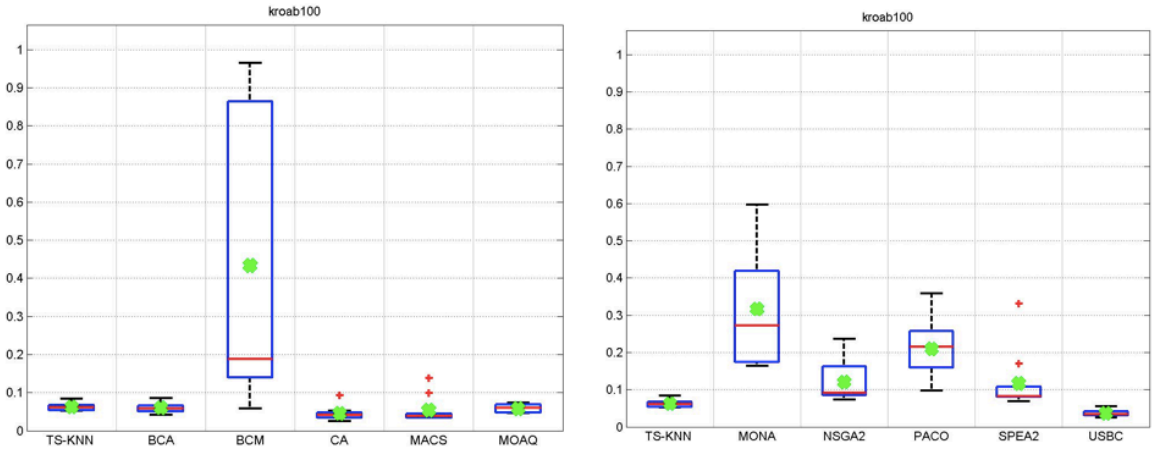

Figura 18. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Espaciamiento con la instancia KROAB100.

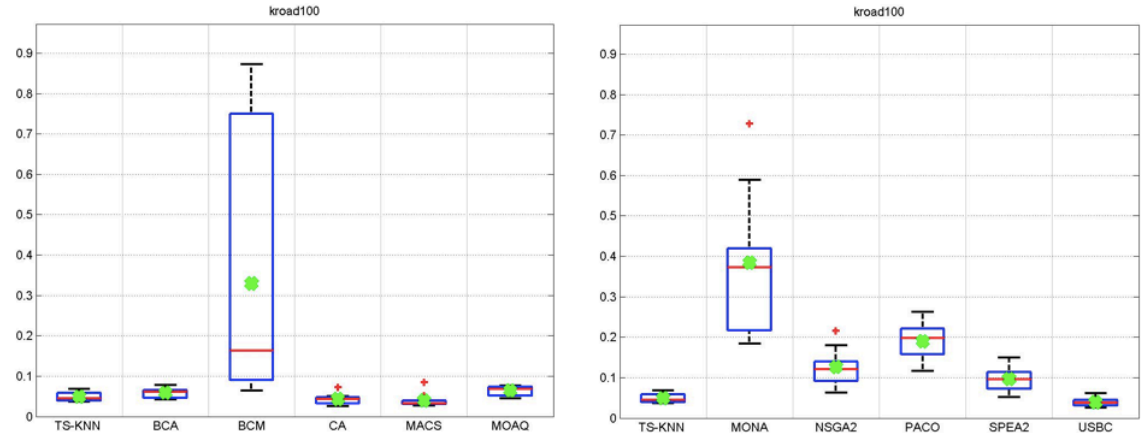

Figura 19. Comparación de diagramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la métrica Espaciamiento con la instancia KROAD100.

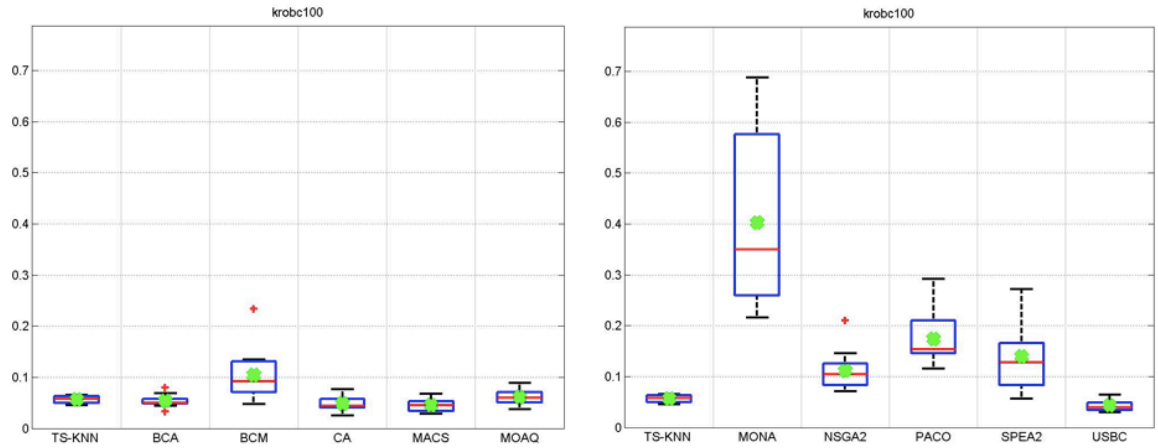

Figura 20. Comparación de digramas de Caja y Bigotes obtenidos por TS-KNN, BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, para la metrica Espaciamiento con la instancia KROBC100.

5. CONCLUSIONES Y TRABAJOS FUTUROS

Luego de haber finalizado la investigación podemos establecer las principales conclusiones:

Se diseñó e implementó una Metaheurística de Optimización Combinatoria (TS-KNN), utilizando el Cluster de ciudades como estrategia para tratar el Problema del Agente Viajero Multi-objetivo, que permite la aproximación de soluciones al problema combinatorio con dos objetivos.

Asimismo, se hizo la revisión del estado del arte para el problema del agente viajero o TSP (Por sus siglas en inglés), teniendo como conclusión que la propuesta es pertinente por la implementación de una estrategia de cluster para el enfoque multi-objetivo del TSP.

Por último, al contrastar los resultados obtenidos por la metaheurística TS-KNN con las soluciones de los algoritmos BCA, BCM, CA, MACS, MOAQ, MONA, NSGA2, PACO, SPEA2 y USBC, bajo las mismas condiciones de experimentación, se puede evidenciar que, TS-KNN presenta un mejor conjunto de soluciones no dominadas en la mayoría de los casos, o en su defecto logra dominar en direcciones en las que el resto de algoritmos no lo hacen.

Por otro lado, se proponen los siguientes trabajos a futuro:

Explorar el uso de heurísticas como reemplazo a Tabú Search, y el uso de otros métodos como reemplazo al Método del Vecino más Cercano.

Realizar pruebas intensivas sobre los valores de α (direcciones en el frente de Pareto) cercanos a 0 y 1 para identificar la razón por la que las soluciones en estas direcciones tienen un mayor espaciamiento.

Además, proponemos realizar pruebas sobre los valores de α cerca de 0.5, debido a que en esta dirección, el algoritmo PACO siempre dominó nuestras soluciones.

6. REFERENCIAS

Aarts, E. Lenstra, J. (1997). Local Search in Combinatorial Optimization. New York, USA. Jhon Wiley & Sons, LTD.

Ahmed, Z. H. (2014). The ordered clustered travelling salesman problem: A hybrid genetic algorithm. The Scientific World Journal, 2014.

Ariyasingha, I. D. I. D., & Fernando, T. G. I. (2016, December). A modified Pareto strength ant colony optimization algorithm for the multi-objective optimization problems. In Information and Automation for Sustainability (ICIAfS), 2016 IEEE International Conference on (pp. 1-6). IEEE.

Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms

Bellman, R. (1962). Dynamic programming treatment of the travelling salesman problem. Journal of the ACM (JACM), 9(1), 61-63.

Bjorklund, A. (2010, October). Determinant sums for undirected hamiltonicity. In Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on (pp. 173-182). IEEE.

Chisman, J. A. (1975). The clustered traveling salesman problem. Computers & Operations Research, 2(2), 115-119.

Christofides, N. (1976). Worst-case analysis of a new heuristic for the travelling salesman problem (No. RR-388). Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group.

Cockbaine, J. Silva, R. (2013). Perfeccionando algoritmos heurísticos para el problema NP-C E-TSP. Ingeniare: Revista Chilena de Ingeniería, ISSN 0718-3291, Vol. 21, Nº. 2, 2013, págs. 196-204

Cunquero, R. (2003). Algoritmos heurísticos en optimización combinatoria. *Universidad de Valencia, Facultad de Ciencias Matemáticas*.

Deb, K. (2001). Multi-Objetive Optimization Using Evolutionary Algorithms. New York, USA. Jhon Wiley & Sons, LTD.

De Klerk, E., & Dobre, C. (2011). A comparison of lower bounds for the symmetric circulant traveling salesman problem. Discrete Applied Mathematics, 159(16), 1815-1826.

Desport, P. Lardeux, F. Lesaint, D. Liret, A. Di Cairano-Gilfedder, C. Owusu, G. "Model and Combinatorial Optimization Methods for Tactical Planning in Closed-Loop Supply Chains," 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), San Jose, CA, 2016, pp. 888-895.

Du, D. Pardalos, P. (1998). Handbook of Combinatorial Optimization. Kluwer Academy Publisher.

Dumitrescu, A., & Mitchell, J. S. (2001, January). Approximation algorithms for TSP with neighborhoods in the plane. In Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms (pp. 38-46). Society for Industrial and Applied Mathematics.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis II. An analysis of several heuristics for the traveling salesman problem. SIAM J. Comput., 6(3):563–581, 1977

Ergun, Ö., & Orlin, J. B. (2006). A dynamic programming methodology in very large scale neighborhood search applied to the traveling salesman problem. Discrete Optimization, 3(1), 78-85.

Española, R. A. (1952). Real Academia Española. Castalia.

Ferreira, J. C., Steiner, M. T. A., & Guersola, M. S. (2017). A Vehicle Routing Problem Solved Through Some Metaheuristics Procedures: A Case Study. IEEE Latin America Transactions, 15(5), 943-949.

Gallart, J. (2009). Análisis, Diseño e Implementación de un Algoritmo Meta Heurístico Grasp que permita resolver el Problema de Rutas de Vehículos con Capacidad (Tesis de pregrado). Pontificia Universidad Católica del Perú. Lima, Perú.

García-Martínez, C., Cordón, O., & Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research, 180(1), 116-148.

Gerhard, R. (1991). TSPLIB—A Traveling Salesman Problem Library. ORSA Journal on Computing 19913:4 , 376-384

Glover, F., & Laguna, M. (1997). Tabu Search. Boston, USA. KLUWER ACADEMIC

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. Journal of the Society for Industrial and Applied Mathematics, 10(1), 196-210.

Hernández-Pérez, H., & Salazar-González, J. J. (2004). A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery. Discrete Applied Mathematics, 145(1), 126-139.

Jaszkiewicz, A. (2002). Genetic local search for multi-objective combinatorial optimization. European journal of operational research, 137(1), 50-71.

Kaufman, L., & Rousseeuw, P. J. (2009). Finding groups in data: an introduction to cluster analysis (Vol. 344). John Wiley & Sons.

Karp, R. M. (1975). On the computational complexity of combinatorial problems. Networks, 5(1), 45-68.

Krolak, P., Felts, W., & Marble, G. (1971). A man-machine approach toward solving the traveling salesman problem. Communications of the ACM, 14(5), 327-334.

Levine, M. S. (2000). Finding the right cutting planes for the TSP. Journal of Experimental Algorithmics (JEA), 5, 6.

Li, X., & Liu, J. B. (2017). A novel approach to speed up ant colony algorithm via applying vertex coloring. International Journal of Parallel, Emergent and Distributed Systems, 1-10.

Lokin, F. C. J. (1979). Procedures for travelling salesman problems with additional constraints. European Journal of Operational Research, 3(2), 135-141.

Mestria, M., Ochi, L. S., & de Lima Martins, S. (2013). GRASP with path relinking for the symmetric Euclidean clustered traveling salesman problem. Computers & Operations Research, 40(12), 3218-3229.

Mestria, M. (2016). A Hybrid Heuristic Algorithm For The Clustered Traveling Salesman Problem. Pesquisa Operacional, 36(1), 113-132.

Mora, A. M., García-Sánchez, P., Merelo, J. J., & Castillo, P. A. (2013). Pareto-based multi-colony multi-objective ant colony optimization algorithms: an island model proposal. Soft Computing, 17(7), 1175-1207.

Niño Ruiz, E. (2010). Diseño e Implementación de una Metaheurística Basada en Autómatas Finitos Deterministas para la Optimización Multiobjetivo de Problemas Combinatorios. (Tesis de Maestría Inédita). Universidad del Norte. Barranquilla, Colombia.

Niño Ruiz, E. (2012). Optimización Combinatoria: Una perspectiva desde la teoría de autómatas. EAE Editorial Academia Espanola.

Morales, J. (2016). Un Algoritmo Memético para la Optimización en Paralelo del TSP (Tesis de Maestría). Instituto Tecnológico de La Paz, La Paz, Mexico.

Mulvey, J. M., & Crowder, H. P. (1979). Cluster analysis: An application of Lagrangian relaxation. Management Science, 25(4), 329-340.

Ozden, S. G., Smith, A. E., & Gue, K. R. (2017). Solving large batches of traveling salesman problems with parallel and distributed computing. Computers & Operations Research, 85, 87-96.

Papadimitriou, C. Steiglitz, K. (2013). Combinatorial Optimization. Prentice Hall.

Park, H. S., & Jun, C. H. (2009). A simple and fast algorithm for K-medoids clustering. Expert systems with applications, 36(2), 3336-3341.

Pérez, J. Jaramillo, G. Sacrificio cortoplacista adaptativo en comparación metaheurísticas para el TSP. Avances en Sistemas e Informática, [S.l.], v. 8, n. 3, p. 125-138, sep. 2011. ISSN 1909-0056.

Potvin, J. Y., & Guertin, F. (1996). The clustered traveling salesman problem: A genetic approach. In Meta-Heuristics (pp. 619-631). Springer US.

Prasad, D., & Mukherjee, V. (2016). A novel symbiotic organisms search algorithm for optimal power flow of power system with FACTS devices. Engineering Science and Technology, an International Journal, 19(1), 79-89.

Shigenobu, R. Furukakoi, M. Yona, A. Senjyu, T. "New optimization method considering combinatorial and multi-objective optimization problem for distribution systems," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 2656-2661.

Tsai, C. F., Tsai, C. W., & Tseng, C. C. (2004). A new hybrid heuristic approach for solving large traveling salesman problem. Information Sciences, 166(1), 67-81.

Van Veldhuizen, D. A., & Lamont, G. B. (2000). On measuring multiobjective evolutionary algorithm performance. In Evolutionary Computation, 2000. Proceedings of the 2000 Congress on (Vol. 1, pp. 204-211). IEEE.

Wang, Y. (2014, August). A Nearest Neighbor Method with a Frequency Graph for Traveling Salesman Problem. In Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on (Vol. 1, pp. 335-338). IEEE.

Yuan, Y. Ong,Y. Gupta, A. Tan, P. Xu, H. "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP," 2016 IEEE Region 10 Conference (TENCON), Singapore, 2016, pp. 3157-3164.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation, 8(2), 173-195.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm.

Zitzler, E., & Thiele, L. (1998, September). Multiobjective optimization using evolutionary algorithms—a comparative case study. In international conference on parallel problem solving from nature (pp. 292-301). Springer, Berlin, Heidelberg.