UNIVERSIDAD DE PANAMÁ

VICERRECTORÍA DE INVESTIGACIÓN Y POSGRADO

FACULTAD CIENCIAS NATURALES, EXACTAS Y TECNOLOGÍA

PROGRAMA REGIONAL DE DOCTORADO EN MATEMATICA

CONSEJO SUPERIOR UNIVERSITARIO CENTROAMERICANO

UN PROBLEMA DE DISPERSIÓN CON CAPACIDADES

IRIS MARINA JIMÉNEZ HIDALGO

TESIS PRESENTADA COMO REQUISITO PARA OPTAR POR EL GRADO DE DOCTORADO EN MATEMÁTICA

DIRECTORES:

Dr. RAFAEL MARTÍ CUNQUERO

Dr. JUANJO PEIRÓ RAMADA

PANAMÁ, REPÚBLICA DE PANAMÁ

2020

Título de la Tesis:

"Un problema de dispersión con capacidades"

TESIS

Sometida para optar al título de Doctorado en Matemática

Vicerrectoría de Investigación y Postgrado

Consejo Superior Universitario Centroamericano (CSUCA)

Facultad de Ciencias Naturales, Exactas y Tecnología

APROBADO POR:

Dr. Rafael Martí
Presidente

Dr. Joaquín Pacheco
Miembro

Dr. José Laguardia
Miembro

REFRENDADO POR:

REPRESENTANTE DE LA VICERRECTORÍA
DE INVESTIGACIÓN Y POSTGRADO

FECHA: 13 de noviembre de 2020.

DEDICATORIA

Gracias a Dios por permitirme tener y disfrutar a mi familia; a mis padres, Dora y Juan José, por ser los principales promotores de mis sueños, gracias por la confianza que siempre han tenido en mí; por el apoyo en cada proyecto que he emprendido y por los consejos que me sirvieron de inspiración y guía en mi vida.

A mis hermanos Giovanna y Juan José por su apoyo incondicional, pero sobre todo por producir a mi alrededor energía positiva.

AGRADECIMIENTO

Iniciar un proyecto como lo es una tesis doctoral, implica el desarrollo de habilidades profesionales, académicas y humanas que modifican el carácter, fortaleciendo la voluntad, la constancia y el optimismo; no es un camino fácil desarrollarla; de hecho, se viven momentos de duda, dificultad e incertidumbre, sin embargo, el deseo de superación y la motivación interna y externa se convierten en aliados para enfrentar y superar esos momentos.

Para que una idea pase a ser un proyecto y se convierta en una tesis, un elemento importante es el asesor, en mi caso tuve dos: el Dr. Rafael Martí, científico de primera que lideriza investigaciones en Metaheurística y Optimización Combinatoria, de quien estoy agradecida por haberme dado la oportunidad de recurrir a su capacidad y conocimiento científico; el Dr. Juanjo Peiró quien me brindó buena parte de su valioso tiempo, por haber tenido toda la paciencia del mundo conmigo. Bajo su orientación y guía este proyecto es una realidad.

Mi agradecimiento a la Universidad de Panamá, institución en la cual laboro, por el apoyo institucional proporcionado durante el proceso relacionado con los estudios doctorales.

Un agradecimiento especial a: 1. La Universidad de Burgos, y en especial al Dr. Joaquín Pacheco, director de la escuela de doctorado de la Facultad de Ciencias Económicas y Empresariales, por acondicionar un espacio de trabajo en el que iniciamos este proyecto, 2. A la Facultad de Ciencias Matemática, en Valencia, que me brindó un espacio físico para realizar este proyecto durante mis (dos) pasantías, así como a los profesores del área de Investigación Operativa por su apoyo y palabras de aliento.

Mi reconocimiento y agradecimiento al Consejo Superior Universitario Centroamericano (CSUCA), entidad generadora y promotora del Programa de Doctorado en Matemática para Centroamérica, que me permitió participar en este proyecto tan importante y necesario para toda la Región Centroamericana.

Y para finalizar, un agradecimiento especial al Dr. José Del Rosario Garrido, profesor titular de la Universidad de Panamá, quien está a cargo de la Coordinación del Doctorado auspiciado por el CSUCA, por su fe y confianza en mí y por ser mi mayor motivador en la realización de este proyecto.

RESUMEN

Los problemas de dispersión y diversidad surgen de la inquietud de encontrar las mejores ubicaciones para instalaciones no deseadas, administración de personal y en el contexto de las redes sociales, entre otros. Maximizar la diversidad se ocupa, en términos generales, de seleccionar un subconjunto de elementos de un conjunto dado de tal manera que se maximice la distancia entre los elementos seleccionados.

En este trabajo presentamos una variante del problema de la Dispersión con restricciones de capacidad y costos dado por **Rosenkrantz, Tayi y Ravi (1999)**, que muestra una heurística con garantía de desempeño de 2, lo que significa que, en todos los casos, el valor de la solución óptima dividido por el valor de su solución es inferior o igual a 2. Para realizar esta variante, investigamos la adaptación de las metodologías de búsqueda adaptativa aleatoria codificada (GRASP) y la búsqueda de entornos variables (VNS) al problema de dispersión con capacidades (CDP), con el objetivo de proponer una hibridación entre GRASP y VNS que se implementará mediante un procedimiento de oscilación estratégica, que nos ayudará a encontrar un método competitivo en la búsqueda de soluciones de alta calidad al problema (CDP).

Para evaluar el rendimiento de nuestra propuesta, realizamos una extensa experimentación para establecer primero los parámetros clave de búsqueda de la heurística y luego compararlos con el método anterior. Además, proponemos un modelo matemático para obtener soluciones óptimas para instancias de pequeño tamaño y comparar nuestras soluciones con el conocido software Local-Solver.

ABSTRACT

Dispersion and diversity problems arise from the concern to find the best locations for undesired facilities, personnel management, and social networks, among others. Maximizing the diversity is in general terms, about selecting a subset of elements from a given set in such a way that the distance between the selected elements is maximized.

In this paper we present a variation of the Dispersion problem with capacity and cost constraints presented by Rosenkrantz, Tayi and Ravi (1999), who propose a guarantee performance heuristic of 2, which means that, in all cases, the value of the optimal solution divided by the value of its solution is less than or equal to 2. To conduct this variant, we investigated the adaptation of the greedy randomized adaptive search (GRASP) and the variable neighborhood search (VNS) to the capacitated dispersion problems (CDP), with the goal of proposing a hybridization between GRASP and VNS to be implemented by a strategic oscillation procedure, which will help us to design a competitive method in the search for high-quality solutions to the problem (CDP).

To evaluate the performance of our proposal, we conducted an extensive experimentation to establish first the key search parameters of the heuristic and then compare them with the previous method. Furthermore, we propose a mathematical model to obtain optimal solutions for small instances and compare our solutions with the well-known Local-Solver software.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción

El problema de la Dispersión de Instancias forma parte de los problemas de Optimización Combinatoria. Resolverlos tiene implicaciones en la vida real, sobre todo cuando se quieren ubicar distintos servicios y estos deben considerar diversos criterios como lo son la cercanía a otros lugares, los costos de ubicación, la capacidad de almacenamiento de estas ubicaciones y alguna que otra característica relacionada con la actividad que se desarrolle en dichas localidades.

De acuerdo con **Bosque y Franco (1995),** la localización de instalaciones de distintos tipos de servicios produce dos tipos de efectos en la población: uno positivo (instalaciones de escuelas, hospitales, comercios, etc.) y otro negativo (instalaciones de vertederos de basura, centrales nucleares, cárceles, etc.), por lo que clasifican las instalaciones de servicios en dos grandes tipos: a) las *deseables* y b) las *no deseables*. Estas últimas son motivo de nuestro estudio.

Según **Bosque Sendra, Gomez Delgado, Moreno Jiménez Y Dal Pozzo (2000)**, los principios básicos a considerar para la localización óptima de instalaciones *no deseables* serían los conceptos concernientes a **Eficiencia Espacial** (*búsqueda de locaciones* que resulten beneficiosa de manera general perjudicando lo menos posible a las poblaciones y sus alrededores) y **Justicia Espacial** (*grado de equidad* entre riesgos y molestias generadas en la población por la ubicación de instalaciones no deseables). Además, estos autores plantean que hay que tener en cuenta, de acuerdo con la eficiencia espacial, dos elementos importantes; por un lado, la mayor lejanía de estas instalaciones de la población directamente afectada y al mismo tiempo la mayor cercanía posible a los productores y usuarios de la instalación.

En el estudio de la ubicación de las instalaciones *no deseables*, existe una amplia literatura al respecto. En **Erkut y Newman (1989),** hacen una recopilación de los modelos estudiados antes de 1989 sobre el problema de la localización de instalaciones *no deseables*. Su investigación se centra en modelos de maximización cuya función objetivo involucra distancias. Proponen un esquema de clasificación para los problemas de localización de instalaciones **no deseables**, atendiendo a ciertos criterios que pueden ser utilizados en otros problemas de ubicación. Algunos de estos criterios son:

a. Número de instalaciones a ubicar: una instalación o varias instalaciones.
b. Interacciones: Interacciones entre población y nuevas instalaciones y entre nuevas instalaciones. Problemas cuantitativos y cualitativos.
c. Objetivos: Un objetivo; multiobjetivo.
d. Región factible: discreta; continua.
e. Restricciones de distancia: límite superior, límite inferior, sin restricciones.
f. Espacio solución: Problemas unidimensionales o multidimensionales; discretos o continuos; restringidos o no restringidos.

Los diferentes modelos que se han propuesto para tratar el problema de la ubicación de instalaciones *no deseables* requieren de una forma de medir la diversidad, donde se considera una función de distancia (entre instalaciones y la población), cuya definición es personalizada de acuerdo con las aplicaciones específicas. De acuerdo con **Bosque et al (2006)**, estos modelos son herramientas para la toma de decisión que se relacionan con los principios de eficiencia y equidad. Presentamos a continuación un resumen de estos modelos:

a) Modelo MaxiSum, o anti-mediano: de acuerdo con este modelo se maximiza la distancia entre las instalaciones y la población.

b) Modelo MaxiMin: el objetivo de este modelo es maximizar la distancia mínima entre los centros poblados y la(s) instalación(es) no deseable(s) más próxima(s).

c) Modelo anti-cobertura: con este modelo se establece un radio de cobertura que indique las áreas donde se generan las molestias y perjuicios graves que provocan la instalación de ubicaciones no deseables. La solución de este modelo minimiza la población afectada que se ubicada dentro de ese radio de cobertura.

Si además de considerar la distancia entre localidades, se toma en cuenta las capacidades de almacenamiento de cada localidad, estaríamos ante un problema de dispersión con capacidades que pertenece al conjunto de problemas NP-Hard, por lo que no se conocen algoritmos eficientes que encuentren soluciones óptimas al mismo; es por ello que no es común en la práctica que estos problemas se resuelvan de forma exacta, sino que se buscan soluciones de calidad, lo suficientemente buenas en un tiempo corto. Para lograr esto, usualmente se utilizan técnicas Metaheurísticas.

Objetivo General y Objetivos Específicos de la Investigación

Objetivo General

Analizar el problema de la Dispersión con restricciones de capacidad y proponer un procedimiento llamado Oscilación Estratégica que alterne fases constructivas y destructivas y que sirvan como base para crear un método competitivo, que busca obtener soluciones de alta calidad al problema.

Objetivos Específicos

1. Investigar la literatura relacionada con el Problema de la Dispersión con Capacidades (CDP) en cuanto al análisis de estos problemas, los modelos relacionados, las formulaciones matemáticas y las técnicas de resolución.
2. Investigar sobre las técnicas heurísticas GRASP y VND y su eficiencia en la solución de problemas de optimización combinatoria.
3. Desarrollar una formulación matemática eficiente para el CDP.
4. Integrar mediante una hibridación los métodos GRAPS y VND en una estrategia de oscilación para obtener soluciones de calidad en el CDP.

Metodología

La investigación que presentamos es de tipo experimental, complementadas con aspectos teóricos que dan sustento a los algoritmos y métodos desarrollados. A través de esta investigación se propone un algoritmo híbrido para el estudio del problema de Dispersión con Capacidades desarrollando un método conocido como estrategia de oscilación, el cual nos permitirá obtener soluciones de calidad.

En el desarrollo de esta investigación, se han seguido los siguientes pasos:

1. Revisión bibliográfica de revistas, textos, apuntes y material virtual relacionado con el tema de la Dispersión de Localidades No deseables.
2. Formulación matemática del problema de Dispersión con Capacidades.
3. Verificación de la formulación matemática del problema mediante la aplicación del CPLEX (software comercial que nos da soluciones óptimas exactas), considerando

diversas instancias, con el fin de analizar la relación instancias versus tiempo para obtener soluciones óptimas.

4. Analizar el algoritmo previo de la Dispersión con Capacidades propuesto por **Rosenkrantz, Tayi y Ravi(1999)**, algoritmo motivador de esta investigación.
5. Diseño de un algoritmo que combina GRASP para la construcción de soluciones y VND como heurística aplicada a búsquedas locales implementando con ello una estrategia de oscilación que nos debe llevar a obtener soluciones de calidad para el problema de Dispersión con Capacidades.
6. Calibración de los parámetros utilizados en el diseño de algunas funciones mediante la experimentación computacional.
7. Implementación de programas computacionales al algoritmo diseñado, utilizando diferentes tamaños de instancias para verificar el comportamiento del algoritmo propuesto al confrontarlo con el algoritmo previo.
8. Comparación del algoritmo propuesto con softwares comerciales como CPLEX y LocalSolver con el fin de establecer la rapidez y la eficiencia del algoritmo propuesto.

Desarrollo de la Investigación

El contenido de la presente investigación se ha organizado de la siguiente manera:

1. Se presenta una breve introducción sobre localización de ubicaciones, específicamente sobre ubicaciones *no deseable,* marco de referencia de nuestra investigación.
2. Se describen los objetivos de la investigación además de la forma en que se llevó a cabo la investigación.
3. En el capítulo 1, se hace referencia a las investigaciones que se han realizado sobre el Problema de la Dispersión, en especial la relacionada con El problema de la Dispersión con restricciones de capacidad y costos presentado por **Rosenkrantz, Tayi y Ravi (1999)**. En este capítulo también se hace referencia a la formulación matemática del problema de la dispersión considerando capacidades: en las localidades y una capacidad requerida por el sistema en estudio.

4. El capítulo 2, trata sobre los métodos que se utilizarán para resolver el problema, específicamente se hace referencia a las heurísticas GRASP, a un método de búsqueda por vecindades conocida como VND; estas dos formando un híbrido tal que en el proceso de mejora del GRASP, en su búsqueda local, utiliza el VND; y a una estrategia que nos ayuda a obtener soluciones de buena calidad llamada Estrategia de Oscilación.
5. El capítulo 3, describe como las heurísticas antes mencionadas se utilizan en el diseño de un algoritmo eficiente para resolver el problema de la dispersión con capacidades, motivo de esta investigación.
6. El capítulo 4, presenta la experimentación computacional de este algoritmo y la calibración de ciertos parámetros involucrados en el mismo, además se hacen comparaciones con un algoritmo previo llamado T1 y con los resultados que se obtienen al aplicar los softwares comerciales CPLEX y LOCALSOLVER, utilizando para esta comparación instancias de distintos tamaños. Además, y a manera de ilustración, presentamos un problema de aplicación para el problema de la dispersión, relacionado con una propuesta para la ubicación de centros de capacitación en un nuevo proyecto de interés social que implica el mejoramiento habitacional y la creación de una nueva comunidad

CAPÍTULO 1

EL PROBLEMA DE LA DISPERSIÓN

El problema de la dispersión con capacidades es un problema NP-hard, que pertenece a la familia de los problemas de dispersión o diversidad. Al consultar la literatura relacionada con los modelos de dispersión, observamos que esos estudios se han centrado en maximizar la diversidad sin tener en cuenta, en su mayor parte, la introducción de restricciones.

Según **Curtin y Church (2006)**, uno de los objetivos que definen a la ciencia de la ubicación es maximizar la dispersión. Las instalaciones *no deseables* se pueden dispersar, para una amplia variedad de propósitos, incluyendo mantener separados a los competidores del mismo sistema de franquicias, dispersar las instalaciones de rehabilitación criminal de los centros de población, la ubicación de bases militares, ya que en el caso en que se dé un ataque, no las destruyan todas al mismo tiempo, y ubicar las plantas de tratamiento de aguas residuales y de energía nuclear de manera que se maximice la seguridad. Este problema involucra la selección de un número específico de lugares de un conjunto en general (lugares que deberán ser los mejores en términos de ubicación), con el fin de maximizar la mínima distancia entre los sitios seleccionados.

El modelo más estudiado de esta clase de problemas es el Problema de la Diversidad Máxima (MDP, Maximum Diversity Problem), en el cual se maximiza la suma de las distancias entre los elementos seleccionados, es decir,

$$\max_{M \subset V, |M|=m} \sum_{i<j, i,j \in M} d_{ij} x_i x_j$$

Donde,V ={1,2,….., n} es el conjunto de todos los sitios considerados, $M \subset V$ el conjunto de los m elementos seleccionados de V, d_{ij} distancia entre los elementos i, j y $x_i = \begin{cases} 1 & si\ el\ elemento\ i\ es\ seleccionado \\ 0 & de\ lo\ contrario \end{cases}$

En la literatura el MDP es también conocido, como el modelo de la Diversidad Max-Sum como es establecido en (**Ghosh**, 1996), como el problema de la Máxima Dispersión (Maximum Dispersion) tal como se lo define en (**Wang**, 2009), como el problema de la jorga

con máximo peso en las aristas (Maximum Edge Weight Clique Problem) definido en **(Alidaee, 2007)**, entre otras denominaciones.

Otro modelo muy documentado en la literatura es el denominado Problema de la Diversidad Max-Min (MMDP, Maximum-Minimum Diversity Problem), definido en (Erkut E. y., 1989), en el cual se maximiza la mínima distancia entre los elementos del subconjunto seleccionado, es decir,

$$\max_{M \subset V, |M|=m} \left\{ \min_{i<j, i,j \in M} d_{ij} x_i x_j \right\}$$

Para resolver estos problemas difíciles de optimización combinatoria se han propuesto muchas heurísticas, como las establecidas en (Glover F. K., 1998), y algoritmos basados en meta-heurísticas, como puede verse en (Duarte, 2007).

1.1 Modelos basados en Equidad

Prokopyev et al (2009) propone tres modelos para maximizar la diversidad de un conjunto en el contexto de modelos de equidad. Estos tres modelos son llamados Problema de la Máxima Mínima Suma (Max-MinSum), el Problema de la Máxima Dispersión Promedio (Max-Mean) y el Problema de la Mínima Dispersión Diferencial (Min-Diff).

El problema *Max-Mean* es el problema de optimización que trata de maximizar la diversidad promedio y cuya principal característica que lo hace diferente del resto de los modelos es que el número de elementos seleccionados también es una variable de decisión. Su formulación matemática es la siguiente:

$$max \ \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j}{\sum_{i=1}^{n} x_i}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i \geq 1$$

$$x_i \in \{0,1\}, i = 1, \dots, n$$

El problema *Min-Diff* consiste en minimizar la diferencia entre la mayor dispersión y la menor dispersión; su formulación matemática viene dada por:

$$min\left\{\max_{i,x_i=1}\sum_{j,j\neq i} d_{ij}\,x_j - \min_{i,x_i=1}\sum_{j,j\neq i} d_{ij}x_j\right\}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i = m$$

$$x_i \in \{0,1\}, i = 1,\dots,n$$

Por último, el problema *Max-MinSum* que consiste en maximizar la medida de dispersión de la mínima suma. En términos matemáticos,

$$max\left\{\min_{i,x_i=1}\sum_{j,j\neq i} d_{ij}x_j\right\}$$

Sujeto a:

$$\sum_{i=1}^{n} x_i = m$$

$$x_i \in \{0,1\}, i = 1,\dots,n$$

La formulación matemática de los problemas anteriores indica que son no lineales, por lo que Prokopyev et al (2009) reformulan estos problemas para linealizarlos aplicando las técnicas que aparecen en Glover et al (1987), Wu (1997), Agca et al (2000) y Tawarmalani (2002), introduciendo variables auxiliares y presentando pruebas de equivalencias entre las formulaciones no lineales y las reformulaciones lineales.

Prokopyev et al (2009) en sus experimentos computacionales, analizaron cuatro modelos: Maxsum DP, Maxmin DP, Max-Minsum DP, y Min-Diff DP, utilizando un solucionador como CPLEX 9.0 para aplicar método exacto de solución (100 variables) y GRASP (Feo y Resende, 1995) como método heurístico (por ser problemas de combinatoria muy difíciles (NP-Duros)) y llegaron a la conclusión que algunas medidas coinciden más que otras y que

para lograr un sistema equitativo se debe aplicar una medida equitativa relacionada con el contexto del problema y sus requisitos de mejora.

Martí y Sandoya (2013) apuntan al trabajo de Prokopyev et al (2009) para introducir características de equidad en los modelos de diversidad; hacen un recorrido por los algoritmos heurísticos previos para el problema de la maximización del promedio de la diversidad y muestran como dada una solución parcial con k elementos seleccionados, puede ser complicada utilizando las técnicas heurísticas basadas en la metodología GRASP propuesta por Prokopyev et al (2009). También ponen en relevancia la técnica de linealización que Prokopyev presentó para modelizar esta variante del problema como un problema de programación lineal entera.

Martí y Sandoya (2013) proponen un algoritmo también basado en la metodología GRASP que es aplicado junto a una estrategia de reencadenamiento de trayectorias como un post proceso de optimización. Este post proceso funciona generando un conjunto élite de soluciones obtenidas a través del GRASP aplicado. Las soluciones en el conjunto élite sirven como inicio de la trayectoria del camino de soluciones que se visitarán hasta alcanzar las soluciones destinos. Glover y Laguna (1997) ya mostraron los beneficios de tal exploración de trayectorias, así como Martí y Sandoya muestran en sus experimentos computacionales.

Martí y Sandoya (2013) señalaron que el MMDP refleja mejor la idea de dispersión. Para ilustrar este punto, la Figura 1 muestra las soluciones óptimas de MDP (izquierda) y MMDP (derecha) respectivamente de una instancia con 30 elementos de los cuales seleccionaron 10. Está claro que ambas soluciones tienen una estructura muy diferente, siendo los 10 puntos en la solución MMDP (la de la parte derecha de la Figura 1) Los mejores distribuidos teniendo en cuenta que en la solución de Suma Máxima (que se muestra en la parte izquierda de la figura), podemos encontrar dos puntos que están muy cerca uno del otro.

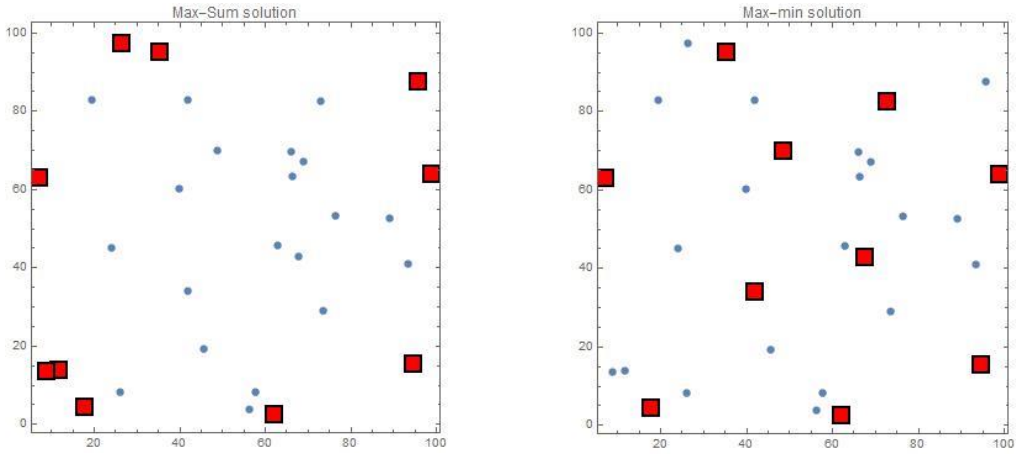

Figura 1. Ejemplo que ilustra las soluciones dadas por el MDP y EL MMDP

1.2 Modelo para el problema de la dispersión con capacidad y costo de almacenamiento para cada sitio.

Rosenkrantz, Tayi y Ravi (1999), presenta varias versiones para el problema de la dispersión con capacidades considerando versiones geométricas y no geométricas. Para identificar estas versiones, Rosenkrantz et al (1999) utiliza el formato sugerido en Garey y Johnson (1979) para nombrar a estos problemas; dicho formato es el siguiente: cada problema tiene un criterio de optimización (Max o Min) seguido de una barra (/) y una o mas medidas de restricción, seguidas también por barras.

a. **Max-Cap/Dist**. Maximizar la capacidad con una restricción de distancia.
b. **Min-Cost/Cap/Dist**. Minimizar costos con restricciones de capacidad y distancia. Dos variantes se presentan para esta versión:
    i. **Max-Dist/Cap/Cost**. Maximizar distancia con restricciones de capacidad y costo.
    ii. **Max-Cap/Dist/Cost**. Maximizar la capacidad con restricciones de distancia y costo.

Para versiones generales del problema de dispersión con capacidades, se tiene:

c. **Max-Cap/Dist**. Maximizar la capacidad con restricción de distancia.
d. **Max-Dist/Cap**. Maximizar distancia con restricción de capacidad. En sus dos versiones:

i. **Sin desigualdad triangular:** de acuerdo con (Ravi et al., 1994), si no cumplen la desigualdad triangular no se puede aproximar a ningún factor en el tiempo polinomial, a menos que P=NP.

ii. **Con desigualdad triangular.** Rosenkrantz, et al (1999) presenta, un algoritmo que basándose en una búsqueda binaria sobre distancias entre vértices, aplica una heurística llamada T1, que tiene una garantía de rendimiento de 2, lo que significa que en todos los casos el valor de la solución óptima dividido por el valor de su solución es menor o igual a 2. Esta versión es la motivadora en el desarrollo de nuestra investigación.

Según Rosenkrantz, et al (1999), consideran a las localidades candidatas como puntos en el plano y la distancia entre cualquier par de localidad como su distancia euclidiana. Establecen una relación entre problemas de dispersión con capacidades y un concepto geométrico denominado el problema del conjunto independiente maximal para una clase especial de grafos, llamados grafos de disco unidad y desarrollan un esquema de aproximación para maximizar la capacidad de almacenamiento total al tiempo que satisface las limitaciones de distancia y costo.

1.3 Formulación Matemática para el CDP

En términos generales el objetivo de estos modelos es determinar la mejor solución (solución óptima) para resolver un problema de decisión donde intervienen un gran número de variables. Esto se logra aplicando técnicas racionales, llámese algoritmos, que consideran la naturaleza de cada problema y los tipos de variables que intervienen en el mismo; identificando, con el desarrollo de estos algoritmos, todas las alternativas de decisión para el problema en estudio y seleccionando aquella que se considere sea la mejor entre las obtenidas.

En algunos problemas de decisión, se requiere que la solución óptima sea un valor entero para algunas de las variables y esto se resuelve estudiando las posibles alternativas de valores enteros para estas variables en el entorno de solución al considerar las variables reales. Los algoritmos que resuelven problemas restringidos a enteros son complejos ya que, a pesar de que el cojunto de soluciones enteras factibles es un conjunto finito, dependiendo del número de variables que intervienen en el problema, resulta impracticable su enumeración explícita

para hallar el óptimo, ya que requieren mucho tiempo computacional. Por ello se han desarrollado una serie de técnicas basadas en la lógica, que a pesar de que no nos garantizan el óptimo, nos ayudan a encontrar buenas soluciones (muchas veces óptimas) en forma eficiente. Estas técnicas se conocen como heurísticos o métodos aproximados que nos llevan a obtener soluciones de calidad con rapidez en el proceso. Una aplicación clásica de la Programación Entera son los problemas de *Localización* donde, en la búsqueda de soluciones se ha desarrollado una serie de técnicas heurísticas que permiten encontrar soluciones satisfactorias en menos tiempos computacionales.

A continuación, se presenta la formulación matemática del Problema de Dispersión con Capacidades, un problema cuadrático binario difícil de resolver, y que transformaremos a un problema Lineal Entero, haciendo algunas adecuaciones al problema original, para facilitar la solución de este.

1.4 El Problema

Dado un grafo $G = (V, E)$, donde V es el conjunto de n nodos y E el conjunto de aristas; sea d_{ij} la distancia entre dos elementos i, j; c_{ij} la capacidad asociada al nodo i y B el total de la capacidad requerida; el problema de Dispersión con Capacidades, CDP, (Capacitated Dispersion Problem por sus siglas en inglés), se puede formular en términos matemáticos, como un problema cuadrático binario de la siguiente manera:

$$Maximizar\ f(x) = min_{i<j} d_{ij} x_i x_j$$

$$sujeta\ a$$

$$\sum_{i=1}^{n} c_i x_i \geq B\,, \qquad x_i \in \{0,1\},\ \ i = 1, \cdots, n$$

Sea $M \subseteq V$, el conjunto de elementos seleccionados y $U = V \setminus M$ como el conjunto de elementos no seleccionados. El CDP consiste en seleccionar un conjunto de elementos $M \subseteq V$ tal que la distancia más pequeña entre cada par de elementos seleccionados sea máxima, mientras que la suma de sus capacidades sea mayor o igual a B. Los elementos del conjunto $M = \{i: x_i = 1\}$, son aquellos que satisfacen la restricción de capacidad y el objetivo será maximizar el valor de la mínima distancia $f(M)$, para una mejor selección de M.

Considerando la formulación anterior, vemos que el problema es no lineal, ya que se tiene el producto de dos variables y el cálculo de un valor mínimo, lo que dificulta su resolución.

Para linealizar el problema utilizaremos los siguientes criterios:

- Para cada $(i,j) \in E$, introducimos la variable auxiliar $y_{ij} \in \{0,1\}$, cuyo significado definimos como:

$$y_{ij} = \begin{cases} 1 \ \ si\ y\ solo\ si\ i,j\ están\ activadas\ simultáneamente, i < j \\ 0\ en\ cualquier\ otro\ caso \end{cases}$$

  En otras palabras, y para nuestra formulación, y_{ij} toma el valor del producto de ambas variables y será 1 sólo si $x_i = x_j = 1$.
- Esto nos permite evitar el uso del producto de las variables $x_i x_j$ en la formulación matemática. Para ello, agregaremos las restricciones (3), (4) y (5), con lo que superamos el problema de la multiplicación de ambas variables. A estas restricciones se les llama restricciones de *consistencia* ya que hacen que x_i y x_j sean consistentes en términos del significado que hay entre ellas.
- Para sustentar la validez de las restricciones antes mencionadas, haremos la siguiente prueba.

  Sea la restricción:

$$y_{ij} \leq x_i \qquad 1 \leq i < j \leq n \qquad (3)$$

  Prueba: En el siguiente cuadro, haremos la validación de la desigualdad, dando a y_{ij} y a x_i lo valores que admiten por ser variables binarias:

y_{ij}	x_i	$y_{ij} \leq x_i$	Validación
0	1	$0 \leq 1$	Verdadero
	0	$0 \leq 0$	Verdadero
1	1	$1 \leq 1$	Verdadero
	0	$1 \leq 0$	Este caso no puede darse pues 1 no es menor o igual a cero.

Esta misma prueba se aplica de manera similar para la restricción

$$y_{ij} \leq x_j \qquad 1 \leq i < j \leq n \qquad (4)$$

Para la restricción (5) verifiquemos su validez, utilizando un cuadro con los valores de x_i, x_j y y_{ij}.

$$x_i + x_j \leq y_{ij} + 1 \qquad 1 \leq i < j \leq n \qquad (5)$$

Prueba:

x_i	x_j	$x_i + x_j$	valor de y_{ij} por (3) y (4)	$y_{ij} + 1$	$x_i + x_j \leq y_{ij} + 1$	Validación
1	0	1	0	1	$1 \leq 1$	Verdadero
0	1	1	0	1	$1 \leq 1$	Verdadero
0	0	0	0	1	$0 \leq 1$	Verdadero
1	1	2	1	2	$2 \leq 2$	Verdadero

Para el cálculo del valor mínimo, definamos una variable $m \in \mathbb{R}_+$ tal que

$m = min_{i<j} d_{ij} y_{ij}$, y se introduce la restricción

$$m \leq d_{ij} y_{ij} + D(1 - y_{ij}) \qquad 1 \leq i < j \leq n \qquad (6)$$

donde el límite superior D sobre los valores de distancia permite la modelación; y tal cual hemos hecho con las anteriores restricciones, verificamos la validez de esta:

y_{ij}	$d_{ij}y_{ij}$	$D(1-y_{ij})$	$d_{ij} y_{ij} + D(1 - y_{ij})$	*Restricción*
0	0	D	0 + D = D	$m \leq D$
1	d_{ij}	0	$d_{ij} + 0 = d_{ij}$	$m \leq d_{ij}$

De acuerdo con los resultados obtenidos en el cuadro anterior, cuando $y_{ij} = 1$ la expresión $m \leq d_{ij}y_{ij} + D(1 - y_{ij})$ se reduce a $m \leq d_{ij}y_{ij}$. Considerando que el cálculo se realiza para cada par de elementos seleccionados $(i, j \in M)$, m toma el mínimo valor de sus distancias. Por otro lado, cuando $y_{ij} = 0$, la expresión (6) queda como $m \leq D$, siendo D un límite superior para los valores de distancia, es decir, es un valor arbitrario mayor que todas las distancias en G. Entonces esta restricción no se activará cuando $y_{ij} = 0$.

Por lo anteriormente descrito, una reformulación lineal entera para el CDP queda de la siguiente manera:

$$Max\ m \tag{1}$$

Sujeto a

$$\sum_{i=1}^{n} c_i x_i \geq B \tag{2}$$

$$y_{ij} \leq x_i \qquad 1 \leq i < j \leq n \tag{3}$$

$$y_{ij} \leq x_j \qquad 1 \leq i < j \leq n \tag{4}$$

$$x_i + x_j \leq y_{ij} + 1 \qquad 1 \leq i < j \leq n \tag{5}$$

$$m \leq d_{ij}y_{ij} + D(1 - y_{ij}) \qquad 1 \leq i < j \leq n \tag{6}$$

$$y_{ij} \in \{0,1\} \qquad 1 \leq i < j \leq n \tag{7}$$

$$x_i \in \{0,1\} \qquad 1 \leq i \leq n \tag{8}$$

Ilustramos, utilizando esta formulación, cómo se obtienen soluciones óptimas. Consideremos el siguiente ejemplo formado con 50 puntos y con valores de capacidad entre 1 y 1000 para cada punto.

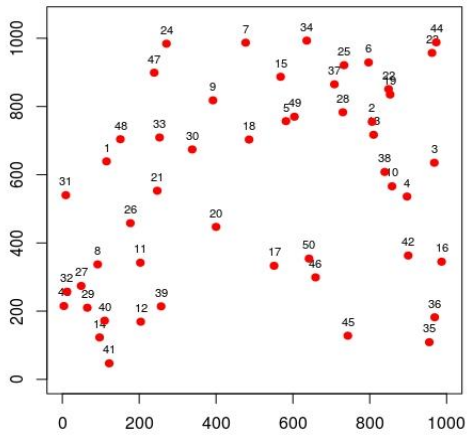

Figura 2. Ejemplo para ilustrar las soluciones de CDP

Para resolver el CDP, utilizaremos el software CPLEX, el cual se ejecutará utilizando la formulación lineal entera, obtenida arriba. En particular, consideremos la capacidad B = 6275. En menos de dos segundos, obtenemos la solución óptima $M = \{1,4,23,24,27,36,46,49\}$, con valor para la función objetivo de m = 331.29 y un valor en la capacidad de 6411. La siguiente figura muestra la selección de puntos en M.

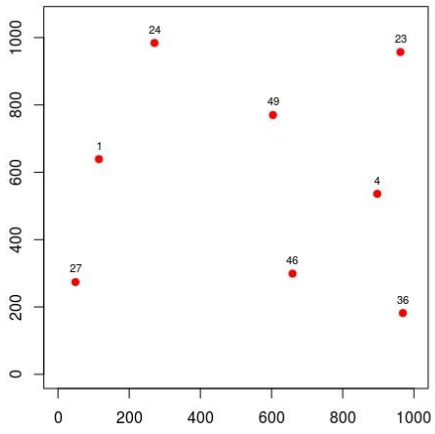

Figura 3. Solución óptima del CDP

Para otros problemas o ejemplos con números de vértices a partir de 100 no hemos conseguido que el software utilizado proporcione una solución de este problema de optimización. Esto se produce por el fenómeno de la explosión combinatoria, pues el espacio de soluciones es inmenso y no se conocen algoritmos de propósito general eficientes.

Por este motivo creemos conveniente utilizar algoritmos y estrategias Metaheurísticas para tratar de resolver ejemplos e instancias más cercanas a los tamaños de los problemas que se pueden encontrar en la vida real.

CAPíTULO 2

MÉTODOS PROPUESTOS PARA RESOLVER EL PROBLEMA DE DISPERSIÓN CON CAPACIDADES

En esta sección, se proponen tres métodos para obtener buenas soluciones para el CDP. El primero se basa en la metodología GRASP el cual se complementa con un método VND como un optimizador de búsqueda local. Finalmente, ambos métodos están integrados en un esquema de oscilación estratégica para obtener mejores resultados.

Antes de describir estos tres métodos, se presentan algunos conceptos relacionados con la optimización combinatoria.

2.1 Optimizacion Combinatoria

La Optimización Combinatoria es una rama de la optimización en matemáticas aplicadas y en ciencias de la computación, que se apoya en:

a. La investigación de operaciones a través de técnicas y métodos analíticos que ayudan a tomar decisiones (su objetivo es la mejor) en la solución de problemas complejos.
b. Teoría de algoritmos, considerando una secuencia de instrucciones ordenadas, lógicas y finitas para obtener buenas soluciones.
c. Teoría de la complejidad computacional, clasificando los problemas de acuerdo con su dificultad computacional, cuantificándolos de acuerdo con la cantidad de recursos que se necesitan para resolverlos como lo son tiempo y memoria.

Los problemas relacionados con la optimización combinatoria son aquellos que intentan dar respuesta a un tipo general de problemas donde se desea elegir el mejor entre un conjunto de elementos y en el que las variables de decisión son enteras y el espacio de solución es discreto o se puede reducir a un conjunto discreto. Cuando hablamos de espacio de solución discreto, consideramos que el óptimo se podría alcanzar mediante la enumeración de todas las soluciones posibles; sin embargo, esta posibilidad se ve restringida a problemas de tamaño pequeños.

En su forma más simple, el problema equivale a resolver una ecuación de este tipo:

$$Max\ (min)\ f(x)$$

$$x \in S \subseteq R^n$$

Donde $x = (x_1, x_2, \ldots, x_n)$ es un vector y representa variables de decisión, *f(x)* es llamada función objetivo y representa o mide la calidad de las decisiones (usualmente números enteros o reales) y S es el conjunto de decisiones factibles.

Algunas veces es posible expresar el conjunto de restricciones como solución de un sistema de igualdades o desigualdades. En un problema de optimización, las restricciones significan que no cualquier decisión es posible.

El objetivo de la Optimización combinatoria es hallar una mejor solución entre un número finito de soluciones viables, y tomar una decisión óptima para maximizar (ganancias, velocidad, eficiencia, etc.) o minimizar un criterio determinado (costos, tiempo, riesgo, error, etc). Este objetivo se encuentra con la dificultad de que enumerar este conjunto de soluciones resulta prácticamente imposible, incluso considerando problemas de tamaño moderado.

2.2 Resolución de Problemas de Optimización Combinatoria

Los siguientes conceptos son importantes y representan la base para entender y ubicar las soluciones de los problemas de optimización combinatoria.

2.2.1 Espacio de Búsqueda.

El tamaño del espacio de búsqueda, punto clave para la resolución de un problema, depende de nuestra representación e interpretación de este, lo que permite encontrar soluciones correctas a nuestro problema y evitar duplicados en los resultados.

2.2.2 Entorno o Vecindad.

Conjunto de soluciones que pueden ser alcanzados desde la solución actual con una operación simple de modificación llamada movimiento. La mejor solución en un vecindario es un óptimo con respecto a su vecindario.

Si nos centramos en una región N(x) del espacio de búsqueda S, que está "cerca" de algún punto x del espacio, podemos definir N(x) como una vecindad del punto $x \in S$.

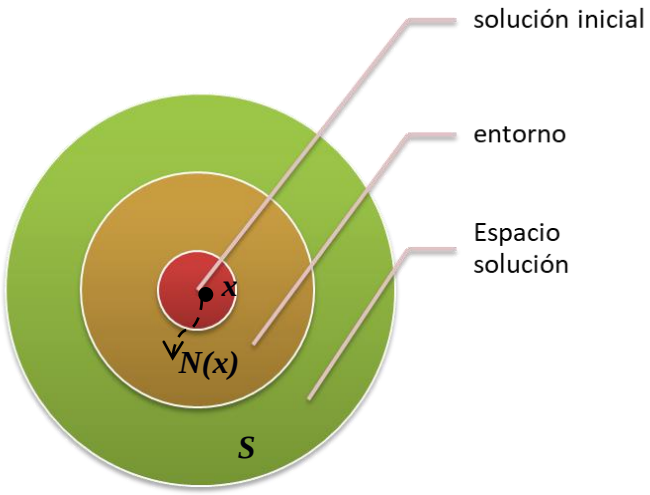

Figura 4. Entorno de x. La flecha indica el movimiento de x en su vecindad en búsqueda de su mejora.

2.2.3 Óptimos Locales y Globales:

La solución x* ϵ S es un ***mínimo local*** del problema si existe un entorno de x*, N(x*) tal que$f(x^*) \leq f(x) \forall x \in N(x^*) \cap S$

La solución x* ϵ S es un ***máximo local*** del problema si existe un entorno de x*, N(x*) tal que$f(x^*) \geq f(x) \forall x \in N(x^*) \cap S$

La solución x* ϵ S es un ***mínimo global*** del problema si $f(x^*) \leq f(x) \forall x \in S$

La solución x* ϵ S es un ***máximo global*** del problema si $f(x^*) \geq f(x) \forall x \in S$

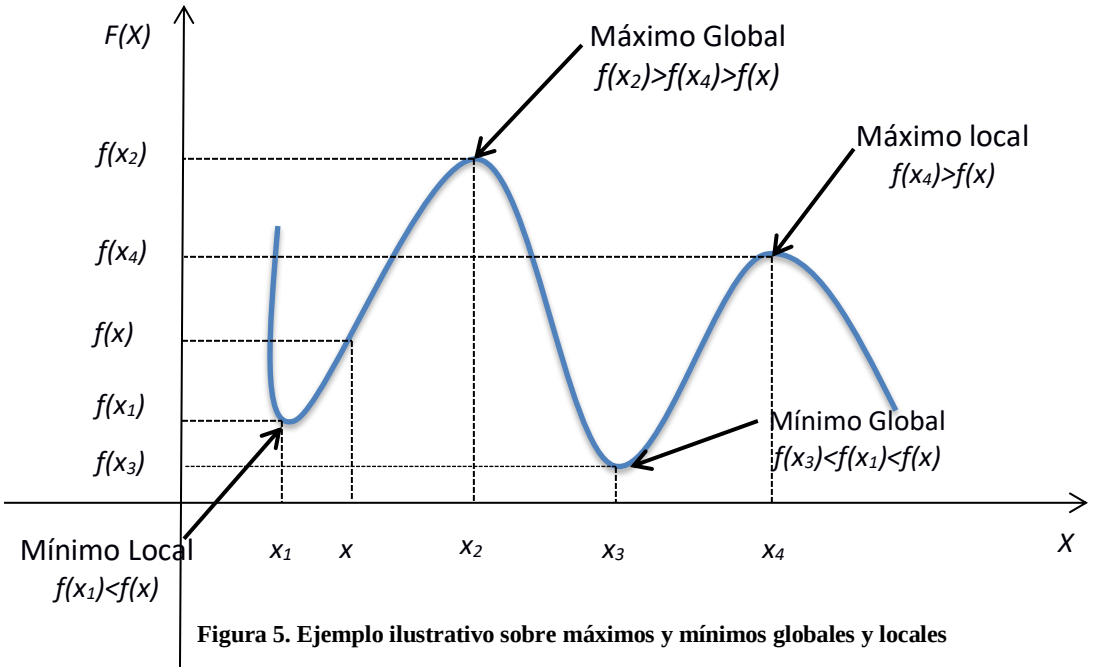

Figura 5. Ejemplo ilustrativo sobre máximos y mínimos globales y locales

2.2.4 Técnicas para la resolución de problemas de optimización combinatoria.

Para encontrar la solución de los problemas de optimización existen diversas técnicas, por ejemplo, tenemos las técnicas exactas (enumerativas, exhaustivas, etc.) que garantizan encontrar la solución óptima de un problema. Sin embargo, a medida que la complejidad del espacio de búsqueda aumenta, el costo de ejecución de dichos algoritmos puede aumentar de forma exponencial, convirtiendo la resolución en prácticamente inviable. Es cuando aparecen las técnicas aproximadas, que sacrifican la garantía de encontrar el resultado óptimo a cambio de obtener una buena solución en un tiempo razonable. Se han venido desarrollando durante los últimos 30 años y se distinguen tres tipos: métodos constructivos, métodos de búsqueda local y las técnicas metaheuristicas.

2.2.4.1 Métodos Constructivos. Suelen ser los más rápidos. Partiendo de una solución vacía, a la que se le van añadiendo componentes, generan una solución completa. Su planteamiento depende en gran parte del tipo de problema. Es muy difícil encontrar métodos de esta clase que produzcan buenas soluciones, y en algunas ocasiones es casi imposible, como, por ejemplo, en problemas con muchas restricciones.

2.2.4.2 Métodos de búsqueda local*:* usan el concepto de vecindario y se inician con una solución inicial completa recorriendo parte del espacio de búsqueda hasta encontrar un óptimo local. En función del operador de movimiento utilizado, el vecindario cambia y el modo de explorar el espacio de búsqueda también, pudiendo la búsqueda complicarse o simplificarse.

2.2.4.3 Algoritmos Metaheurísticos: son algoritmos de propósito general, que no dependen del problema, y que ofrecen buenos resultados pero que normalmente no acaban ofreciendo “la solución óptima” sino soluciones subóptimas. Se utilizan para aquellos problemas en que no existe un algoritmo heurístico específico que los resuelva, o bien cuando no es práctico implementar dichos métodos

2.3 Procedimientos Metaheurísticos en Optimización Combinatoria.

Las técnicas Metaheurísticas son procedimientos que, aunque no garantizan la obtención del óptimo del problema de estudio, se basan en la aplicación de reglas sencillas y fáciles de entender y aplicar. Estas técnicas representan un recurso interesante en problemas de optimización combinatoria, problemas en las que las variables de decisión son enteras (o discretas, al menos), en las que el espacio solución está formado por ordenaciones de los valores de dichas variables.

La forma de ejecución de las Metaheurísticas es similar: parten de una solución (o conjunto de soluciones) que no es óptima y a partir de ella se obtienen otras parecidas, de entre las cuales se elige una que satisfaga algún criterio y se comienza de nuevo el proceso de búsqueda; proceso que culmina cuando se cumple alguna condición de parada establecida previamente.

2.4 Características de las Técnicas Metaheurísticas

De acuerdo con Sadiq, S. M. y Habib, Y., (1999), las técnicas Metaheurísticas poseen las siguientes características:

- Son ciegas, en el sentido de que no saben si llegan a la solución óptima, por lo que se debe establecer criterios de paradas para indicarle cuando detenerse.
- Son algoritmos aproximados, por lo que no garantizan la obtención de la solución óptima.
- En algunos de sus movimientos, aceptan “malos movimientos”, es decir, en el proceso de búsqueda, se encuentran con soluciones que no son necesariamente mejor (de acuerdo con la función objetivo) que la inmediatamente anterior. Este paso se ve como un intermedio que le permite acceder a nuevas regiones no exploradas.
- Son relativamente sencillas ya que sólo se necesita una representación adecuada del espacio de soluciones, una solución inicial (o un conjunto de ellas) y un mecanismo para explorar el espacio de soluciones.
- Son técnicas generales ya que se pueden aplicar en la resolución de cualquier problema de optimización de carácter combinatorio; sin embargo, estas

técnicas serán eficientes si las operaciones que realicen tengan relación con el problema que se plantea.

- La regla de selección dependerá del instante del proceso y del historial que hasta ese momento se tenga. Si en dos iteraciones determinadas, la solución es la misma, en la siguiente iteración la nueva solución no tiene por qué ser la misma.

2.5 Indicadores de calidad de un algoritmo heurístico

Al resolver un problema de forma heurística debemos poder medir la calidad de los resultados ya que la optimalidad no está garantizada. En general un buen algoritmo heurístico debe de tener las siguientes propiedades:

- **Eficiente**. El esfuerzo computacional debe ser realista para obtener la solución.
- **Bueno**. La solución encontrada debe de estar, en promedio, cerca del óptimo.
- **Robusto**. La probabilidad de obtener una mala solución debe ser baja.

2.6 Procedimientos para medir la calidad de un algoritmo heurístico

Comparación con una solución óptima. Con el fin de medir la calidad de la solución obtenida con un método aproximado, se disponen de determinados procedimientos que proporciona el óptimo para un conjunto limitado de ejemplos, generalmente, de tamaño pequeño.

Comparación con una Cota. Algunas veces, el óptimo no está disponible ni siquiera para un conjunto limitado de ejemplos. Una alternativa para medir la calidad de la solución de un heurístico sería comparar el valor de la solución obtenida con el método aproximado, con una cota del problema (inferior si es un problema de minimización o superior si es de maximización). La efectividad de esta medida dependerá de lo buena que resulte la cota (que tan cerca del óptimo esté); por lo que se debe tener cierta información sobre la bondad de esta cota.

Comparación con un método exacto truncado. Un método enumerativo como el de Ramificación y Acotación explora una gran cantidad de soluciones, aunque sea únicamente una fracción del total, por lo que los problemas de grandes dimensiones pueden resultar computacionalmente inabordable con estos métodos. Sin embargo, podemos establecer un

límite de iteraciones (o de tiempo) máximo de ejecución para el algoritmo exacto. También podemos saturar un nodo en un problema de maximización cuando su cota inferior sea menor o igual que la cota superior global más un cierto valor *a* (de manera similar para el caso de minimización). De esta manera, se garantiza que el valor de la mejor solución proporcionada por el procedimiento no dista más de *a* del valor óptimo del problema. En cualquier caso, la mejor solución encontrada con estos procedimientos truncados proporciona una cota con la que comparar al heurístico.

Comparación con otros heurísticos. Este método es el más empleado en problemas difíciles (NP-hard) y sobre los que se han trabajo durante mucho tiempo, por lo que se conocen buenos heurísticos. Lo efectivo de esta comparación, al igual que con la comparación con una cota, es que tan bueno es el heurístico escogido.

A pesar de que las soluciones que ofrecen las técnicas metaheurísticas no son las óptimas, y en general no es posible conocer que tan próximos del óptimo se esté, ellas permiten el estudio de problemas de gran complejidad de una manera sencilla y obtener soluciones suficientemente buenas en tiempos razonables.

2.7 Métodos propuestos para la solución del CDP.

2.7.1 GRASP. Procedimiento Codificado De Búsqueda Adaptativa Aleatoria.

La metodología GRASP (Greedy Randomized Adaptive Search Procedures) por sus siglas en inglés fue propuesta por Feo y Resende (1995) y que en castellano sería como Procedimientos de Búsqueda basados en funciones Voraces Aleatorizadas que se adaptan, en donde cada uno de los términos que conforman el nombre corresponde con las características que desarrolla el método. Este método en su versión básica consiste en iteraciones, donde cada iteración se fundamenta en dos fases: una fase de construcción de una solución factible buena, no necesariamente un óptimo local y una segunda fase donde se aplica un método de mejora para encontrar un óptimo local.

La fase de construcción del GRASP es iterativa, codiciosa, aleatoria y adaptativa.

- ***Iterativa*** porque las soluciones se construyen considerando un elemento en cada paso.

- ***Codiciosa*** porque la adición de cada elemento en la solución está guiada por una función codiciosa que determina la contribución local de cada elemento a la solución parcial; de acuerdo con esto, los mejores candidatos son identificados, creándose una lista restringida de candidatos (RCL).
  - Numéricamente, la RCL suele construirse utilizando las evaluaciones de los valores máximos y mínimos de las características asociadas a los elementos (por ejemplo, coste, distancia, etc.), elementos seleccionables en una iteración dada. Considerando lo anterior, la RCL estaría formada por todos aquellos elementos cuyas evaluaciones superen (para problemas de maximización) el umbral dado por la siguiente expresión:

$$RCL_{umbral} = (evaluación_{min} + \alpha(evaluación_{max} - evaluación_{min}))$$

    donde el parámetro α: $0 \leq \alpha \leq 1$ determina el tamaño de la RCL. Si $\alpha = 1$, en la RCL sólo estaría el mejor candidato (función miope pura), si por el contrario $\alpha = 0$ estarían todos los candidatos (función aleatoria pura). En implementaciones estándares de GRASP el parámetro α se determina de forma aleatoria.

La medida codiciosa es miope ya que no toma en cuenta qué ocurrirá en iteraciones sucesivas al realizar una elección, sino únicamente en esta selección.

- A***daptativa***, ya que los beneficios asociados a los elementos seleccionados son actualizados en cada iteración con el objetivo de que se reflejen los cambios ocasionados por la selección del elemento anterior.
- A***leatoria*** porque no selecciona al mejor candidato según la función codiciosa, sino que los elige al azar de la lista restringida de candidatos con el fin de diversificar y no repetir soluciones en construcciones diferentes.

Como ocurre en muchos métodos, las soluciones generadas por la fase de construcción del GRASP no suelen ser óptimos locales. Dado que la fase inicial no garantiza la optimalidad local respecto a la estructura de entorno en la que se está trabajando (hay selecciones aleatorias), se aplica un procedimiento de búsqueda local como un procedimiento posterior para mejorar la solución obtenida. En este procedimiento de mejora se suele utilizar un intercambio simple con el objeto de no emplear mucho tiempo en esta mejora.

GRASP es un método que se basa en realizar múltiples iteraciones y quedarse con la mejor, por lo que no le resulta especialmente beneficioso detenerse demasiado en mejorar una solución dada. Al realizar muchas iteraciones, GRASP realiza un muestreo del espacio de soluciones, que de acuerdo con observaciones empíricas se ve que la distribución de la muestra generalmente tiene un valor promedio que es inferior al obtenido por un procedimiento determinista, sin embargo, la mejor de las soluciones encontradas generalmente supera a la del procedimiento determinista con una alta probabilidad.

De acuerdo con Feo (1995), una de las características más relevantes del GRASP es su sencillez y facilidad de implementación. Con fijar el tamaño de la lista de candidatos y el número de iteraciones se puede determinar completamente el procedimiento. De esta manera se pueden concentrar los esfuerzos en diseñar estructuras de datos para optimizar la eficiencia del código y proporcionar una gran rapidez al algoritmo, dado que éste es uno de los objetivos principales del método.

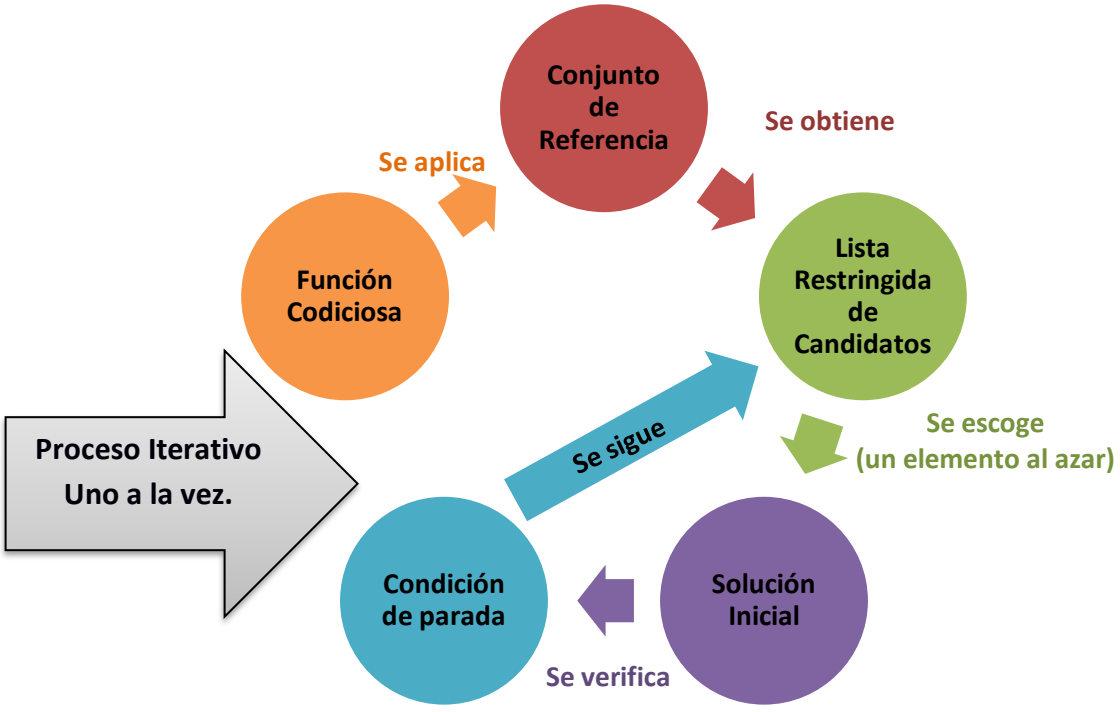

Figura 6 . Proceso de Construcción del GRASP

2.7.2 Búsqueda de Entorno Variables (VNS)

La metodología de Búsqueda de Entornos Variables (VNS por sus siglas en inglés de Variable Neighborhood Search) es una metaheurística que se basa en una idea simple y

efectiva: un cambio sistemático de vecindad dentro de un algoritmo de búsqueda local, cuando la búsqueda se estanca en un óptimo local.

La VNS está basada en tres hechos simples (Hansen, 2003) :

1. Un mínimo local con una estructura de entornos no lo es necesariamente con otra.

2. Un mínimo global es mínimo local con todas las posibles estructuras de entornos.

3. Para muchos problemas, los mínimos locales con la misma o distinta estructura de entornos están relativamente cerca.

Esta última observación, que es empírica, implica que los óptimos locales proporcionan información acerca del óptimo global. Puede ser, por ejemplo, que ambas soluciones tengan características comunes. Sin embargo, generalmente no se conoce cuáles son esas características. Es importante, por tanto, realizar un estudio organizado en las proximidades de este óptimo local, hasta que se encuentre uno mejor.

Los hechos antes mencionados, sugieren el empleo de varias estructuras de entornos en las búsquedas locales para abordar un problema de optimización. El cambio de estructura de entornos se puede realizar de forma determinística, estocástica, o determinística y estocástica a la vez.

Una característica importante de la metaheurística VNS y sus extensiones, es que mantiene simple sus esquemas básicos, permitiendo hacer pocos ajustes en sus parámetros, lo que hace que se generen buenas soluciones con rapidez y de manera muy simple, y que se determine su rendimiento lográndose implementaciones muy eficientes.

2.7.2.1 Estructura de funcionamiento del VNS.

En un problema de optimización se desea encontrar dentro de un conjunto de soluciones factibles, aquella que optimiza una función f(x). Si el problema es de minimización, su formulación sería:

$$\min\{f(x) | x \in X\} \quad \text{(a)}$$

donde x representa una solución alternativa, f es la función objetiva y X es el espacio de soluciones factibles del problema. Una solución óptima x^* (o mínimo global) del problema es una solución factible donde se alcanza el mínimo de (a).

Las Metaheurísticas de búsqueda local aplican una transformación o movimiento a la solución de búsqueda cambiando algunos elementos y por tanto utilizan, explícita o implícitamente, una estructura de entornos, que se define como una aplicación *N*, que asocia a cada solución *x* del espacio *X* un subconjunto de soluciones *N(x)* (entorno de x), cuyas soluciones se dicen vecinas de *x*.

En muchos procesos de búsqueda, se utilizan movimientos fijando o acotando el número de elementos de la solución que se pueden cambiar, obteniéndose los entornos más utilizados en VNS.

Un movimiento sería, por ejemplo, intercambiar ***k*** elementos de la solución, es decir, cambiar ***k*** elementos de la solución por otros ***k*** elementos que no están en la solución, si se trata de soluciones con un número fijo de elementos, o intercambiar la posición de ***k*** elementos de la solución si se trata de ordenamientos. Este último es el tipo de entorno más común en las aplicaciones de la VNS a problemas de optimización.

En Moreno y Mladenović (2003) presentan metaheurísticas basadas en procedimientos de búsqueda local que aplican distintas formas de continuar la búsqueda después de encontrar el primer óptimo local. A continuación, se presentarán algunas.

2.7.2.2 Búsqueda de entorno variable descendente (VND)

La Búsqueda de Entorno Variable Descendente (*VND, Variable Neighborhood Descent*) es una variante de VNS que explora los vecindarios de una manera determinista. En particular, VND explora vecindarios pequeños hasta que se encuentre un óptimo local. En ese punto, el proceso de búsqueda cambia a un vecindario diferente (generalmente más grande) que podría permitir un mayor progreso hacia el óptimo global.

La solución final proporcionada por el algoritmo es un mínimo local con respecto a todas las k_{max} estructuras de entornos, y por tanto la probabilidad de alcanzar un mínimo global es mayor que usando una sola estructura. La mayoría de las heurísticas de búsqueda local usan en sus descensos simplemente un entorno y algunas veces dos ($k_{max} = 2$).

2.7.2.3 Búsqueda de Entornos Variables Básica (BVNS)

El método "básico de búsqueda de entorno variable" (*BVNS Basic Variable Neighbourhood Search*) combina cambios determinísticos y aleatorios de estructura de entornos.

Para la BVNS una condición de parada puede ser, por ejemplo, el máximo tiempo de CPU permitido, el máximo número de iteraciones, o el máximo número de iteraciones entre dos mejoras.

2.7.2.4 Búsqueda de Entornos Variables General

La Búsqueda por Entornos Variables General (GVNS, *General Variable Neighbourhood Search*) combina las búsquedas BVNS con VND al sustituir la busqueda local del paso (1b) de la Busqueda por Entornos Variables basica por una Busqueda por Entornos Variables descendente; es decir, una VND. Esta estrategia utiliza dos series de estructuras de entornos posiblemente distintas, una para la busqueda descendente y otra para los movimientos aleatorios de agitacion.

El uso de la Búsqueda por Entornos Variables general (GVNS) ha dado lugar a las aplicaciones más exitosas aparecidas recientemente.

2.7.3 Estrategia de Oscilación

La estrategia de oscilación (Glover y Hao, 1977), es un procedimiento que fue propuesto en sus inicios para cruzar de un espacio factible a otro no factible. Su éxito radica en su capacidad para integrar la diversificación con intensificación sin recurrir a formas de diversificación aleatorizadas.

En Delgado, 2002, se describe a la estrategia de oscilación como un "proceso que opera orientando los movimientos en relación con una cierta frontera en donde el método normalmente se detendría. Sin embargo, no se detiene, ya que las reglas para la elección de los movimientos se modifican, para permitir que la región al otro lado de la frontera sea alcanzada. Posteriormente, se fuerza al procedimiento a regresar a la zona inicial. El proceso de aproximarse, traspasar, y volver sobre una determinada frontera, crea un patrón de oscilación que da nombre a esta técnica. Una de sus aplicaciones más útiles es crear un proceso de búsqueda expandida, que puede admitir soluciones infactibles durante la búsqueda. Las soluciones infactibles se permiten, pero se penalizan por su grado de infactibilidad.".

La estrategia de oscilación trabaja alternando fases constructivas con fases destructivas, donde cada solución generada por una fase constructiva es destruida por la fase destructiva,

después de lo cual una nueva fase constructiva construye una nueva solución. Los procesos constructivos y destructivos se pueden interrumpir en cualquier punto para aplicar un proceso de mejora. Los procesos se aplican solo al final de una fase constructiva, cuando se obtiene una solución completa, pero se pueden aplicar en otros puntos también. A veces aplicándolos antes en un proceso constructivo, por ejemplo, se pueden eliminar características deficientes que de otra manera serían heredados por etapas posteriores de construcción y que crearía soluciones indeseables.

Para Glover F. y otros (1995) una razón para considerar la estrategia de oscilación al resolver problemas de optimización, está que asegura la diversidad en la búsqueda heurística de soluciones óptimas, mejorando la robustez del método. Al pasar de una región factible a una infactible aplicando los procesos de construcción y destrucción en la generación de soluciones, se consideran estructuras de solución que no son visitadas por el algoritmo de búsqueda. La estrategia de oscilación proporciona un medio bastante efectivo para lograr una interacción entre intensificación (explotación del espacio: cantidad de esfuerzo empleado en la búsqueda en la región actual) y diversificación (exploración del espacio: cantidad de esfuerzo empleado en la búsqueda en regiones distantes del espacio), dos características importantes en cualquier algoritmo de búsqueda de soluciones que balanceadas nos ayuda a obtener buenas soluciones.

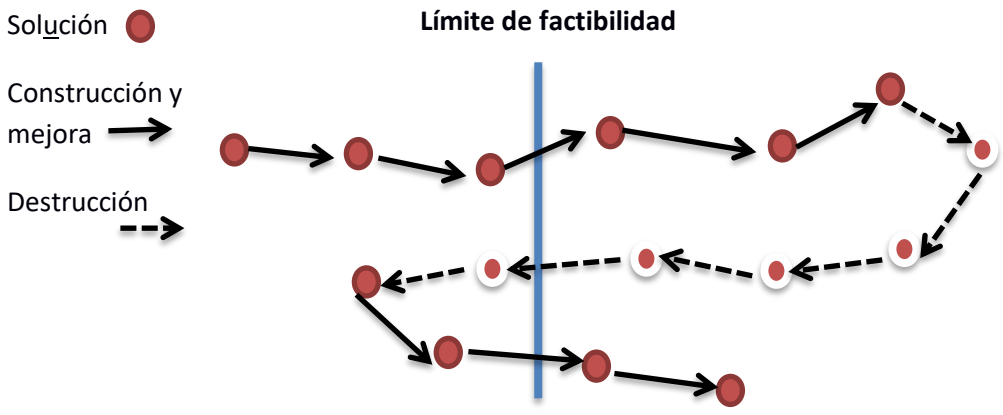

Figura 7. Estrategia de Oscilación. Fases constructivas y Destructivas.

CAPÍTULO 3

EL PROBLEMA DE LA DISPERSIÓN CON CAPACIDADES

3.1 Problema de la Dispersión con Capacidades (algoritmo previo)

Como se mencionó en el capítulo 1, Rosenkrazt et al. (2000) propusieron la única heurística previa conocida para el problema de la dispersión con capacidades. Esta heurística, llamada T1, tiene una garantía de rendimiento de 2, lo que significa que, en todos los casos, el valor de la solución óptima dividido por el valor de su solución es inferior o igual a 2.

3.1.1 T1 heurística

Este método llama a una rutina denominada Greedy_Try con los parámetros α y B, que trata de seleccionar un conjunto de nodos, llamados sitios, de manera codiciosa para satisfacer la restricción de distancia α y la restricción de capacidad B. En este código, $CAP(V')$ denota la suma de las capacidades de los nodos en V'.

La Figura 8 muestra un pseudocódigo de este método anterior.

Heurística T1

1. Ordena los $v_i \in V$ en orden no creciente de sus capacidades y crea una lista llamada Site_List.
2. Ordena las distancias entre los sitios en orden no creciente. Elimina aquellas distancias que se duplican. La lista ordenada se almacenará en un vector D, tal que $D[1] > D[2] > \cdots > D[t]$
3. Realiza una búsqueda binaria sobre el vector D, para encontrar el índice i tal que $\alpha = D[i]$, y al llamar a la función Greedy_Try(α, B), ésta regrese como resultado "éxito" y para $\alpha' = D[i+1]$ al llamar a la función Greedy_Try(α', B) ésta devuelva "error".
4. Devolver la distancia entre vértices α encontrada en el paso 3 y detenerse.

Procedimiento Greedy_Try. Consiste en seleccionar un conjunto de sitios $v_i \in V$ de manera que se satisfagan las restricciones de distancia α y las restricciones de capacidad B.

1. Sea L =Site_List y $V' = 0$
2. Mientras L $\neq 0$
   a. Añadir el primer vértice *v* de L a V'
   b. Quitar de L todos los vértices (incluyendo *v*) tal que $d(v_i, v_j) < \alpha, i < j$
3. Si $CAP(V') \geq B$ entonces devolver "éxito" de lo contrario devolver "fracaso"

Figura 8. Psuedocódigo de la heurística T1

El siguiente ejemplo, que ilustra el funcionamiento del algoritmo, está basado en distancias euclidianas. Tiene cuatro nodos 1, 2, 3 y 4, cada uno con capacidad 2 que están ubicados en los vértices de un cuadrado de dos unidades por lado y un quinto nodo ubicado en el centro del cuadrado con capacidad 1. Aplicamos el algoritmo T1 para resolver el CDP con un límite de capacidad B = 5.

En el paso 1 de T1, creamos Site_List = (1, 2, 3, 4, 5) y en el paso 2 ordenamos las distancias entre vértices, eliminando las que se duplican, obteniendo D = (2.82, 2, 1.41).

Ahora, en el paso 3, llamemos a Greedy_Try (α = 1, B = 5) y realicemos los siguientes pasos:

1. v = 1; L= (4); V' = (1)
2. v = 4; L = 0; V' = (1, 4)
3. Fracaso

Llamemos nuevamente a la función cambiando el valor de α, Greedy_Try (α = 2, B = 5) y realicemos los siguientes pasos:

1. v = 1; L= (2, 3, 4); V' = (1)
2. v = 2; L = (3, 4); V' = (1, 2)
3. v = 3; L= (4); V' = (1, 2, 3)
4. Éxito

Entonces la solución viene dada por V' = (1, 2, 3) con α = 2

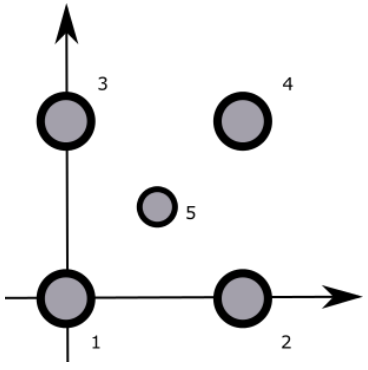

Figura 9. Diagrama para el problema propuesto para el T1

Resolveremos el problema de la figura 2, utilizando la heurística T1. Tal y como se hizo con CPLEX en el desarrollo anterior, consideramos un límite de capacidad B = 6275. Al aplicar la heurística que propone Rosenkrazt, obtenemos la solución

M ={1, 4, 8, 15, 23, 24, 36, 46} con el valor de la función objetivo m = 302.9 y un valor de capacidad B = 6514. La siguiente figura muestra los puntos seleccionados.

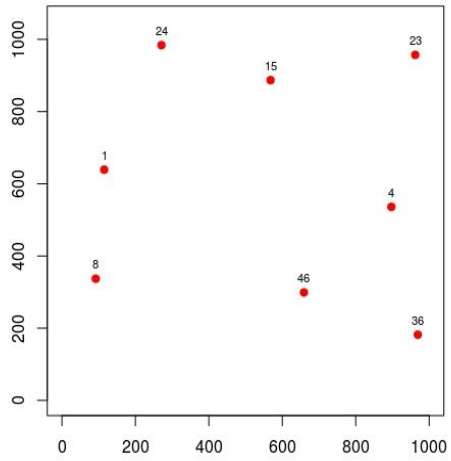

Figura 10. Solución de un problema con 50 nodos, utilizando la heurística T1.

Si comparamos la solución obtenida anteriormente con el solucionador CPLEX, donde se obtuvo un valor de m = 331,29, con la obtenida con la heurística T1, m = 302,9, se concluye que existe un margen de mejora en el diseño y aplicación de un método heurístico.

Este resultado es básicamente la motivación de este trabajo; diseñar un algoritmo heurístico capaz de obtener una solución de alta calidad en tiempo computacional corto.

3.2 Heurísticas Propuesta para el Problema de Dispersión con Capacidades.

Al reformular matemáticamente el CDP, problema de complejidad NP-duro, y utilizando esta formulación para encontrar soluciones óptimas (ejemplo descrito en el capítulo 1) aplicando el software CPLEX, se observa que para instancias con número de vértices mayores de 100, no se ha podido conseguir una solución, ya que el espacio de solución para esta cantidad de vértices es inmenso, por lo que se propone el diseño de una hibridación de las heurísticas GRASP y VNS implementando un procedimiento de Oscilación Estratégica que de soluciones de buena calidad para instancias de mediano y gran tamaño.

3.2.1 GRASP

Para nuestro modelo consideremos el conjunto V con n vértices o nodos, y C1 un procedimiento de construcción que realiza una serie de pasos consecutivos para producir una solución. El conjunto M, inicialmente vacío, representa una solución parcial en esta construcción. En cada paso, C1 selecciona un elemento candidato $i \in V \setminus M$ con una buena evaluación, que será determinada por una función $eval(i)$ que calcula la distancia mínima entre el elemento i y los elementos seleccionados. Luego, C1 construye la lista de candidatos restringidos, RCL, con todos los elementos candidatos (no seleccionados) con una evaluación dentro de una fracción de la evaluación máxima. En término matemático:

$$RCL = \{i \in V \setminus M : eval(i) \geq eval_{min} + \alpha(eval_{max} - eval_{min})\}$$

Donde $\alpha \in [0,1]$ es un parámetro de búsqueda que se ajustará empíricamente. Luego, el método selecciona aleatoriamente un elemento en RCL y lo agrega a la solución parcial M. C1 realiza los pasos siempre que no se cumpla la restricción de capacidad. En otras palabras, el método se detiene cuando la suma de las capacidades de los elementos en M es mayor o igual que B.

Constructivo del GRASP

1. *Condición de parada:*

   Mientras que capacidad_ofrecida < Capacidad dada(B)

2. *Construir una solución parcial M, inicialmente vacía.*
3. *Evaluar los elementos que no estén en la solución*

---

4. *Calcular la función de evaluación*
5. *Construir RCL*
6. *Escoger aleatoriamente un elemento de RCL.*
7. *Introducir en la solución M, el elemento escogido en el paso 7.*
8. *Verificar la condición de parada*

Figura 11. Fase de Construcción del GRASP

Se podría argumentar que C1 es ciego en su proceso de selección con respecto a los valores de capacidad de los nodos. Para superar esta limitación, proponemos C2 con una función de evaluación más elaborada. En particular, primero calcula $d_i = min_{j \in M} d_{ij}$ y $d_{max} = max_i d_i$ y ajusta la contribución del valor de distancia al rango [0,1]. Del mismo modo, calcula $c_{max} = max_i c_i$ y luego:

$$eval(i) = \beta \frac{d_i}{d_{max}} + (1 - \beta) \frac{c_i}{c_{max}}$$

y el peso relativo de estos factores, distancia y capacidad, se ajusta con el parámetro $\beta \in [0,1]$. C2 realiza los pasos de la misma manera que C1, con la única diferencia de la función de evaluación.

Glover (1998) es probablemente el primero en estudiar sobre problemas de diversidad desde una perspectiva de optimización heurística. Los autores anticipan que dado que las diferentes versiones de este problema pueden incluir restricciones adicionales, el objetivo es diseñar la heurística cuyos movimientos básicos para la transición de una solución a otra sean simples y flexibles, permitiendo que estos movimientos se adapten a múltiples configuraciones. Los movimientos que son especialmente atractivos en este contexto son movimientos constructivos y destructivos que impulsan la búsqueda para acercarse y cruzar los límites de viabilidad desde diferentes direcciones. Tales movimientos también son muy naturales en el problema de diversidad máxima, donde el objetivo es determinar una composición óptima para un conjunto de elementos seleccionados. De acuerdo con estas observaciones, proponemos dos métodos "destructivos", llamados D1 y D2.

D1 es la contraparte de C1, en la que, en cada iteración, en lugar de agregar un elemento a la solución parcial, eliminamos un elemento de esta solución parcial. En los métodos destructivos consideramos que inicialmente todos los elementos están seleccionados y eliminamos elementos, uno por uno, siempre que se cumpla la restricción de capacidad. En particular, en cada iteración eliminamos el elemento con una mala evaluación relativa (como lo indica el RCL creado de forma similar a C1 con la misma función de evaluación). De forma similar, D2 es la contraparte de C2 en el sentido de que usa la misma definición de evaluación que C2, pero aquí identifica los elementos malos y anula la selección de uno aleatorio tomado de RCL.

Destructivo del GRASP

1. *Condición de parada*
   *Mientras que λ*capacidad_ofrecida≥Capacidad dada(B)*
2. *Verificar los elementos de la solución M*
   Si capacidad_ofrecida ≥ capacidad dada(B)
   Guardar la solución actual.
   De lo contrario
3. *Evaluar los elementos que están en la solución actual aplicando la función de evaluación*
4. *Construir RCL con candidatos que tengan una mala evaluación relativa*
5. *Escoger aleatoriamente un elemento de RCL*
6. *Eliminar de la solución M, el elemento escogido del paso 5.*
7. *Verificar la condición de parada.*

Figura 12. Fase del destructivo del GRASP

3.2.2 VND

Definimos $N_k(M)\ para\ k = 1,2,\dots,k_{max}$ como el conjunto de soluciones, aquellas que se obtienen al intercambiar *k* elementos en la solución *M* con *k* elementos en *V\M*. Los intercambios en este contexto consisten en reemplazar elementos seleccionados por

elementos no seleccionados. VND se basa en el hecho de que se define un óptimo local con respecto a una relación de vecindario, de modo que si una solución candidata *M* es localmente óptima en un vecindario $N_i(M)$, no es necesariamente un óptimo local para otro vecindario $N_j(M)$. Tengamos en cuenta que, en nuestro problema, debemos verificar que la solución sea factible después del intercambio (es decir, que verifique la restricción (2) en nuestra formulación anterior). Específicamente, cuando se reemplazan *k* elementos en la solución, la suma de las capacidades de los elementos seleccionados después del reemplazo debe ser mayor o igual que *B*. Para simplificar la descripción, sólo consideraremos intercambios factibles.

Dado un conjunto *V* con *n* elementos y una solución factible *M* con algunos de estos elementos seleccionados, calculamos que para cada $i \in M$,

$$dm_i = \min_{j \in M} d_{ij}$$

Tengamos en cuenta que el valor de la función objetivo de esta solución, *f(M)* se puede calcular como el mínimo de los valores dm_i. Es claro que para mejorar una solución necesitamos eliminar (y así reemplazar) los elementos *i* en la solución para los cuales $dm_i = f(M)$. Nuestro método, inicialmente escanea, en cada iteración, la lista de elementos en la solución ($i \in M$) con un valor mínimo dm_i. En particular, escanea la lista de elementos en orden lexicográfico, y para cada elemento i^* con un valor mínimo dm_i, considera la lista de elementos no seleccionados ($j \in V \backslash M$) en la búsqueda del primer intercambio de mejora en $N_1(M)$. Considerando que el conjunto $V \backslash M$ es relativamente grande (comparado con el conjunto *M*), implementamos una estrategia para escanearlo de manera eficiente y encontrar un buen intercambio para i^*. En particular, calculamos $de_j(i^*)$ para $j \in V \backslash M$, donde,

$$de_j(i^*) = \min_{i \in M \backslash \{i^*\}} d_{ij}$$

escanea estos vértices en orden inducido por los valores de d_e, donde el que tiene el valor más grande es el primero. Debe tenerse en cuenta que los valores de d_e son un indicador de la calidad potencial de un elemento para formar parte de la solución cuando eliminamos i^*. Dado que la función objetiva es maximizar los valores entre distancias, cuanto mayor sea d_e, mejor es el elemento. Es por esto que exploramos primero los elementos con grandes valores

de d_e en nuestra búsqueda de un buen intercambio. De hecho, es una estrategia de descenso más pronunciada (Glover y Laguna, 1977).

El método realiza el primer movimiento de mejora y actualiza dm_i para todos los elementos en *M*. En esta primera fase, el algoritmo repite iteraciones siempre que se puedan realizar intercambios de mejora y cuando no sea posible una mejora adicional, recurre a considerar intercambios de dos elementos, implementando de esta manera un VND. Limitamos nuestro método a dos vecindarios, $N_1(M)$ y $N_2(M)$, para evitar los tiempos de ejecución más grandes asociados con los vecindarios grandes.

Un punto importante en los problemas de Max-Min es la definición de movimiento de mejora. Los artículos anteriores sobre este tipo de problemas consideran una definición extendida, que incluye no solo cuando el movimiento aumenta el valor de $f(M)$, sino también cuando mejora un determinado indicador (ver, por ejemplo, Resende et al. 2010). En nuestro algoritmo VND vamos a probar ambas posibilidades: un criterio estándar de mejora basado simplemente en $f(M)$, y un criterio extendido en el cual el número de elementos $i \in M$ para el cual $dm_i = f(M)$ se reduce, también consideramos que la solución mejora. Vamos a comparar ambos diseños en el estudio experimental en la Sección 3.3.

Como en las implementaciones típicas de VND, nuestro método atraviesa la lista de estructuras de vecindario de forma secuencial (es decir, forma 1 o 2). En particular, *k* se establece inicialmente en 1; luego, en cada paso, se determina un vecino *M'* de *M* que se mejora en $N_k(M)$: si *M'* es mejor que *M* de acuerdo con la definición extendida anterior, entonces *M* se reemplaza con *M'*; de lo contrario, *k* se incrementa en una unidad (es decir, $k = k+1$). En otras palabras, el algoritmo realiza una búsqueda local para mejorar la solución en $N_1(M)$ y recurrir únicamente a $N_2(M)$ cuando la búsqueda está atrapada en un óptimo local encontrado en $N_1(M)$. Siguiendo la estrategia llamada Basic VND (Duarte et al. 2018), cuando se implementa una jugada de mejora y se actualiza la solución existente, el método regresa al primer vecindario (es decir, $k = 1$). Finalmente, cuando se ha explorado todo el vecindario y no se encuentra ningún movimiento de mejora ($k=2$), el método VND se detiene.

3.2.3 SO. Oscilación Estratégica

La Oscilación Estratégica (Glover y Laguna, 1997) opera al orientar los movimientos en relación con un nivel crítico, que en nuestro problema se define como el nivel de capacidad. Tal nivel crítico o límite de oscilación a menudo representa un punto donde el método normalmente se detendría. Sin embargo, en lugar de detenerse cuando se alcanza este límite, las reglas que seleccionan movimientos (en el método constructivo C1 y el destructivo D1) se modifican para permitir que se cruce la región definida por el nivel crítico. Luego, la aproximación avanza para una profundidad específica más allá del límite de oscilación y da la vuelta. El límite de oscilación nuevamente se aproxima y se cruza, esta vez desde la dirección opuesta, y el método avanza hacia un nuevo punto de retorno. El proceso de aproximación y cruce repetidos del nivel crítico desde diferentes direcciones crea un comportamiento de oscilación, que da nombre al método.

El método constructivo C1 descrito en la sección 3.2.1 agrega elementos, uno por uno, a la solución actual en construcción, hasta que la suma de las capacidades de los elementos seleccionados sea mayor o igual que B. En este punto, si se tiene una solución factible, por ejemplo, M_1, el método se detiene. Lo que proponemos ahora es "seguir adelante" unos pocos pasos más. En particular, consideramos la adición de γ elementos adicionales a la solución M_1, con el mismo método C1, obtención de la solución M_2. Luego, aplicaremos un método destructivo, como D1, para eliminar de M_2 algunos elementos. De esta manera, podemos obtener una nueva solución factible M_3, que eventualmente podría ser mejor que las dos anteriores.

De acuerdo con la estrategia "seguir adelante" descrita con anterioridad, al agregar elementos, proponemos eliminar algunos elementos adicionales de la solución factible. En particular, aplicamos D1 para eliminar γ elementos adicionales de M_3, obteniendo así una solución parcial e inviable M_4. Si repetimos este esquema, agregando ahora elementos a M_4 con C1, obtenemos una nueva solución factible M_5, lo que nos da un patrón de oscilación que cruza el límite de factibilidad del espacio de solución, como se ilustra en la figura 12.

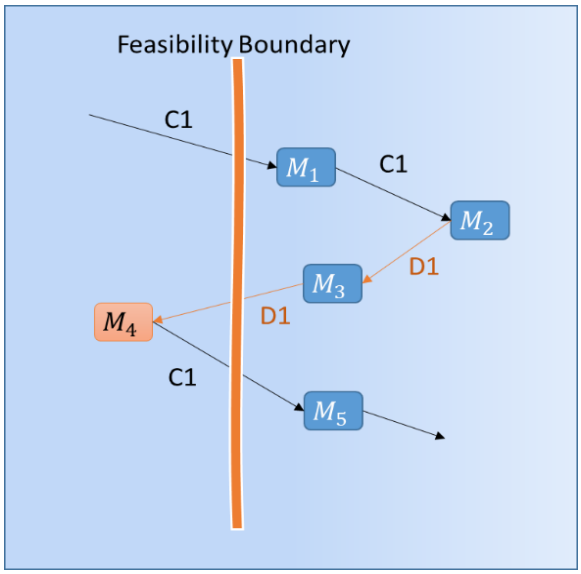

Figura 13. Patrón Estratégico de Oscilación

En nuestra estrategia de oscilación, agregamos y eliminamos vértices de acuerdo con el límite de capacidad. Específicamente, al aplicar C1 para agregar vértices adicionales a la solución, que corresponde a los pasos para pasar de la solución M_1 a M_2, figura 12, multiplicamos la suma de las capacidades de los elementos en la solución mediante un factor $\lambda \in [0,1]$ y detenerse cuando este producto es más grande que B. En términos matemáticos, aplicamos C1 mientras que $\lambda \sum_{i \epsilon M} c_i \geq B$.

Proponemos una selección del método VND para mejorar algunas de las soluciones, en lugar de aplicarlas a todas las soluciones encontradas en el proceso. En particular, si consideramos el "camino de las soluciones" cuando aplicamos C1 de la "región no factible" a la "región factible", podemos identificar la primera solución factible. Eso correspondería a la solución M_1 y M_5 en la figura 12. Aplicamos VND a estas soluciones, y no lo aplicamos al resto de las soluciones en el camino obtenido con C1 (es decir, omitimos M_2 y M_4 a la aplicación de VND). Simétricamente, al aplicar el método destructivo D1 de la "región factible" a la "región no factible", identificamos la última solución visitada antes de abandonar la región factible. Eso sería M_3 en la figura 12. Aplicamos VND a estas últimas soluciones. De esta manera, ahorramos tiempo de cómputo y tratamos de evitar obtener los mismos óptimos locales al aplicar VND desde diferentes soluciones iniciales.

CAPÍTULO 4

EXPERIMENTOS COMPUTACIONALES

En esta sección se describen los experimentos computacionales realizados para calibrar y evaluar los métodos utilizados en esta investigación. Se establecen en primer lugar los valores de los parámetros que intervienen en los métodos desarrollados; se comparan las soluciones obtenidas al desarrollar el algoritmo propuesto SO y las obtenidas en el método heurístico previo T1, con las soluciones óptimas dadas por CPLEX al resolver el problema de dispersión con capacidades. Además, se calculan las soluciones con LocalSolver para las instancias de prueba utilizadas. En ambos solucionadores, CPLEX y LocalSolver, se implementa el modelo matemático presentado en el capítulo 1.

4.1 Instancias utilizadas en el Problema

Para la experimentación, se utilizó el dominio público MDPLIB (Martí et al., 2019) disponible en http://grafo.etsii.urjc.es/optsicom, que contiene varios conjuntos de datos empleados anteriormente en diferentes estudios sobre problemas de diversidad (Sandoya et al., 2018). Se revisaron esos casos y se adaptaron a la versión con capacidades del problema de diversidad. En particular, para cada instancia original, se generaron de manera aleatoria el valor de capacidad de cada nodo en el rango [1, 1000]. Luego, se calculó la suma de todas las capacidades y se estableció a B (total de la capacidad requerida) como esta suma multiplicada por un factor de 0,2 y 0,3 respectivamente, creando así dos instancias para cada una original. La referencia utilizada para el problema de diversidad con capacidades consta de las siguientes 100 instancias.

GKD: este conjunto de datos, propuesto originalmente por Glover et al. (1998), contiene matrices para las cuales los valores se calcularon como las distancias euclidianas desde puntos generados aleatoriamente con coordenadas en el rango de 0 a 10. Este conjunto contiene tres subconjuntos de instancias:

> GKD-a: Glover et al. (1998) introdujo las pequeñas instancias en este conjunto con valores de $n \leq 30$. No se consideran estos casos porque son demasiado pequeños.

GKD-b: Martí et al. (2010) generó estas instancias de tamaño mediano con valores de $25 \leq n \leq 150$. Para el CDP se consideraron 10 instancias de tamaño 50 y 10 de tamaño 150 en este conjunto, y se generaron dos instancias para cada una de ellas como se describió anteriormente.

GKD-c: Duarte y Martí (2007) generaron estas instancias grandes con $n = 500$. Se consideraron 10 instancias en este conjunto y se generaron dos de ellas con diferentes valores de capacidad como en los otros conjuntos.

SOM: este conjunto de datos consta de 70 matrices con números aleatorios entre 0 y 9 generados a partir de una distribución uniforme de enteros. Martí et al. (2010) creó estas instancias para resolver el problema de diversidad máxima, en el que la función objetivo es la suma de las distancias. Como en el CDP, la versión que se maneja es que la función objetivo se calcula con la distancia mínima, se encontró que la mayoría de estas instancias no se pueden usar, ya que muchos puntos están a una distancia de 0, lo que hace que la función objetivo sea 0. Se seleccionaron 10 instancias en el subconjunto SOM-a con $n = 50$ que puede usarse para el CDP. Como en los conjuntos anteriores, se generan dos conjuntos de instancias para la versión de capacidad con los factores 0,2 y 0,3.

MDG: este conjunto de datos fue generado por Duarte y Martí (2007) y se utilizó en Gallego et al. (2009) y Palubeckis (2007). Se compone de 100 matrices con números reales seleccionados al azar de una distribución uniforme.

MDG-a: Este conjunto contiene instancias con números reales en el rango 0, 10. No se considera porque no son adecuados para los problemas Max-Min, ya que algunos puntos están a una distancia de 0.

MDG-b: Este conjunto contiene instancias con números reales en el rango 0, 1000. Se consideran 10 instancias con $n = 500$, para lo cual se crean dos de ellas con el factor de capacidad establecido en 0,2 y 0,3 como se describe anteriormente.

La tabla 1 resume las 100 instancias de referencia, utilizadas en CDP.

Nombre	*N*	Capacidad Paramétrica	Número de instancias
GKD-b2	50, 150	0,2	20
GKD-b3	50,150	0,3	20
GKD-c2	500	0,2	10
GKD-c3	500	0,3	10
SOM-a2	50	0,2	10
SOM-a3	50	0,3	10
MDG-b2	500	0,2	10
MDG-b3	500	0,3	10

Tabla 1. Conjuntos de instancias.

4.2. Configuración del algoritmo y ajustes

4.2.1 Ajustes de los Parámetros α, β, γ

En la parte experimental de la investigación es importante encontrar configuraciones efectivas para los métodos utilizados en el CDP, por lo que se hace necesario ajustar los parámetros algorítmicos, asociados con estos métodos.

En la heurística GRASP, en su fase constructiva se consideran los parámetros (*α)* para C1 y (α, β) para C2; en la fase destructiva aparece el parámetro (γ) para D1 y (γ, β) para D2 y además se considera otro parámetro (λ) para SO. Todos estos parámetros están en el rango [0, 1].

Para encontrar los valores de estos parámetros que mejor se ajustaban a las características de los métodos aplicados a la investigación, se analizaron un conjunto de 24 instancias representativas del conjunto de 100 descrito anteriormente (Tabla 1). Para cada experimento, se presentaron las siguientes medidas de rendimiento:

1. *f(M),* número promedio de valor mínimo de inter-distancia

2. *Time(s):* tiempo de cálculo en segundos
3. *Dev:* promedio relativo de la desviación porcentual con respecto a la mejor solución encontrada en el experimento,
4. *Best:* número de mejores soluciones encontradas en el experimento.

Considerese que tanto Dev como *Best* se refieren a las soluciones encontradas dentro del experimento y no a las mejores soluciones conocidas para estos problemas. La desviación relativa se calcula para cada instancia como la diferencia del mejor valor obtenido en este experimento menos el valor obtenido con cada algoritmo particular, dividido por el mejor valor. Al multiplicar este resultado por 100 se tiene la representación en porcentaje.

En este primer experimento preliminar, se prueba el método constructivo C1 descrito en la Sección 3.2.1. El rendimiento de C1 depende del parámetro **α**, que equilibra la codicia y la aleatoriedad. Se prueban cuatro valores de **α** en cada procedimiento (0,2, 0,4, 0,6 y 0,8). La tabla 2 muestra los resultados de este experimento.

Procedimiento(α)	*f(M)*	*Dev (%)*	*Best*	*Time(s)*
C1(0,2)	32	5,8	4	<1
C1(0,4)	32	3,4	6	<1
C1(0,6)	33	2,1	9	<1
C1(0,8)	33	6,3	9	<1

Tabla 2. Comparación de diferentes valores α en el método constructivo C1.

El valor de desviación mínimo se obtiene con **α = 0,6**, que también puede obtener el mayor número de las mejores soluciones (9). Por lo tanto, se selecciona este valor para establecer **α** en C1.

En el segundo experimento, se exploran los diferentes valores de los dos parámetros en el método constructivo C2 (α, β). Como en C1, el parámetro **α** gestiona la composición de la lista de candidatos restringida (RCL), en este segundo experimento se prueban valores relativamente grandes para este parámetro, favoreciendo la codicia en la composición de RCL. En particular, se consideran dos valores: 0,6, 0,8. Por otro lado, el parámetro β es el

peso relativo entre la distancia y la capacidad en el proceso de selección. Toma valores en [0,1] y cuánto más cerca está el valor de 1, más importante es la distancia con respecto a la capacidad. Se prueban los valores 0,6, 0,8 para β, puesto que la distancia juega un papel importante en el problema. La tabla 3 muestra los resultados de este experimento.

Procedimiento (α, **β)**	$f(M)$	*Dev (%)*	*Best*	*Time (s)*
C2(0,6, 0,6)	58,5	6,8	5	<1
C2(0,6, 0,8)	54,0	17,2	3	<1
C2(0,8, 0,6)	57,7	7,7	6	<1
C2(0,8, 0,8)	54,8	17,5	4	<1

Tabla 3. Comparación de los diferentes valores de α y β en el método constructivo C2.

La desviación mínima en la Tabla 3 se obtiene con α = β = 0,6, que muestra una desviación del 6,8 por ciento. Se seleccionan este valor para estos dos parámetros.

En los experimentos siguientes, se explora el rendimiento de los métodos destructivos D1 y D2 tal y como se hizo con C1 y C2. Se hace un análisis similar al de las tablas 2 y 3 y se llega a la conclusión es que el mejor desempeño se logra en D1 cuando γ = 0,4 y en D2 con **β = 0,6, γ = 0,5.** Estos valores se utilizarán en los siguientes experimentos.

4.3 Rendimiento de las Metaheurísticas GRASP, VND y SO

Una pregunta importante en la búsqueda heurística se relaciona con el origen de las soluciones de alta calidad obtenidas después de aplicar el método de mejora (VND en la investigación). Es de interés medir la correlación entre el valor de las soluciones construidas y el valor de las soluciones mejoradas (es decir, los óptimos locales obtenidos con VND). Una pregunta interesante sería: ¿Los óptimos locales de alta calidad provienen de construcciones de alta calidad?

Para responder a la pregunta anterior, se generan 100 soluciones con C2 para cada problema en el conjunto de entrenamiento y se calcula el coeficiente de correlación entre la solución inicial y el óptimo local obtenido con VND. El coeficiente de correlación resultante es 0,5, que es relativamente bajo, lo que indica que los óptimos locales buenos pueden provenir de soluciones iniciales de cualquier calidad. Se realiza un experimento adicional, el quinto,

para probar este punto. En particular, la Figura 17 muestra estos dos valores de función objetivo, el valor construido (C2 en el eje x) y el valor mejorado (VND en el eje y), para cada una de las 100 soluciones generadas en una instancia representativa en la prueba. Se observa que los puntos están dispersos en el plano sin ningún patrón discernible, lo que ilustra esta falta de correlación.

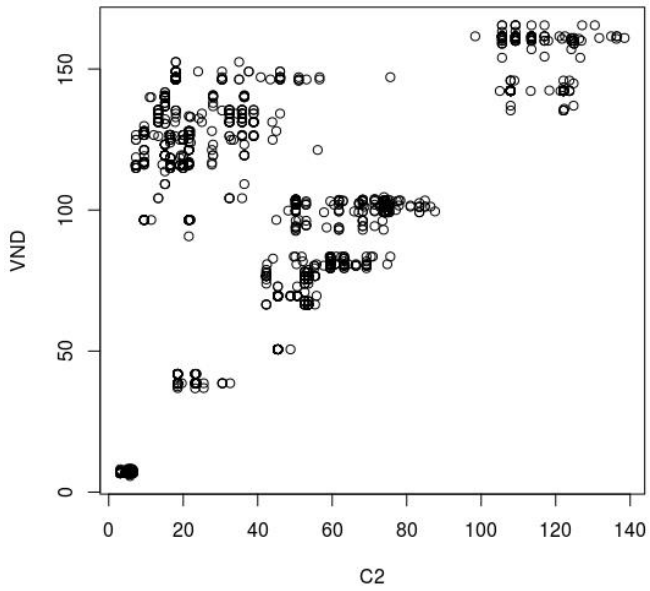

Figura 14. Diagrama de dispersión que representa valores de soluciones construidas y mejoradas.

Ahora se procede a comparar los cuatro métodos propuestos para generar soluciones cuando se combinan con el método de mejora VND. Se consideran estos métodos con sus parámetros de búsqueda clave ajustados en los experimentos anteriores. La Tabla 4 muestra los resultados del sexto experimento preliminar para revelar el mejor método de generación. Al igual que en el experimento anterior, se consideran las siguientes cuatro estadísticas para compararlas: *f(M), Dev., Best y Time*(en segundos).

Procedimiento	$f(M)$	***Dev (%)***	***Best***	***Time (s)***
C1(0,6)+VND	97,4	3,7	8	5151,7
C2(0,6,0,6)+VND	99,4	2,8	7	11,4
D1(0,4)+VND	98,7	3,8	2	10,1
D2(0,6,0,5)+VND	97,7	4,3	2	8,7

Tabla 4. Comparación de los diferentes métodos GRASP con VND.

El mejor método constructivo en la Tabla 4 es C2 (0,6, 0,6) + VND con una desviación de 2,8 por ciento y 7 mejores soluciones, mientras que el mejor método destructivo es D1 (0,4) + VND con una desviación de 3,8 por ciento y 2 mejores soluciones. Por lo tanto, estos métodos se utilizarán en la heurística de Oscilación Estratégica (SO) completa, en la que se incorporan métodos constructivos y de mejora.

Como se describe en la Sección 3.2.3, el SO funciona aplicando primero un método constructivo en el que se realizan iteraciones adicionales más allá del límite factible, como lo indica el parámetro **λ**. Luego, se aplica un método destructivo para "volver" a ese límite y volver a cruzarlo. La alternancia de ambos métodos establece el patrón de oscilación que da nombre a la metodología. Se denotará a este método híbrido como SO (λ). La Tabla 5 muestra los resultados cuando se comparan diferentes valores de λ. Para simplificar los datos de la tabla, se presenta la desviación porcentual promedio (Dev) y el tiempo de ejecución en segundos (Tiempo). En la tabla también se incluyen diferentes valores para el número de oscilaciones.

	Número de Oscilaciones									
	1		2		3		4		5	
λ	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**	**Dev**	**Time**
0,6	5,4%	37	1,3%	69	0,2%	131	0,5%	139	0,7%	168
0,7	5,3%	35	2,1%	71	3,5%	106	2,5%	148	2,3%	141
0,8	5,2%	36	3,9%	55	4,6%	89	4,0%	131	3,9%	137
0,9	5,4%	43	4,5%	68	4,6%	72	4,0%	145	4,1%	167

Tabla 5. Método de oscilación estratégica.

La Tabla 5 muestra la evolución de la desviación de la mejor solución encontrada con el método SO. Por ejemplo, en la primera fila, que corresponde a λ = 0,6, se ve que después de la primera oscilación, el método obtiene una solución con un 5,4% de desviación en 37 segundos (esta sería la solución M_4 en la Figura 12). Luego, en la segunda oscilación, el método es capaz de mejorar esta solución, obteniendo una nueva solución con 1,3% de desviación, lograda en 69 segundos. Continuando con el proceso de oscilación, luego de 5

oscilaciones, se ve que el método obtiene una solución con un 0,7% de desviación lograda en 168 segundos. Al comparar las diferentes filas en esta tabla, se concluye de que $\lambda = 0,6$ obtiene los mejores resultados y, por lo tanto, y se establece el valor de este parámetro para los siguientes experimentos.

4.4 Pruebas competitivas

Para las pruebas competitivas, comparamos el procedimiento que desarrollamos, SO, con diferentes métodos para probar su rendimiento relativo, tanto en términos de tiempo de ejecución como de calidad. En particular, comparamos SO, establecido con C2 (0,6, 0,6) + VND como proceso constructivo, D1 (0,4) + VND como proceso destructivo, y $\lambda = 0,6$ como valor de oscilación, con los siguientes métodos:

- La heurística anterior para este problema, T1, por Rosenkrantz et al. (2000)
- El solucionador CPLEX, que proporciona las soluciones óptimas para casos pequeños
- El localSolver heurístico de propósito general.

Este experimento consiste en ejecutar los tres procedimientos anteriores en el conjunto completo de 100 instancias en nuestro punto de referencia y comparar sus resultados con los obtenidos con SO. Presentamos los resultados en 3 tablas, una para cada método respectivamente. La tabla 6 muestra los resultados obtenidos con SO y T1. Al igual que en los experimentos preliminares, establecemos el valor de la función objetivo, ***f (M)***; la desviación del porcentaje relativo con respecto a la mejor solución en este experimento, ***Dev***; el número de instancias en cada conjunto en el que el método puede coincidir con la mejor solución conocida, ***Best***, y el tiempo de ejecución en segundos, ***Time***. Cada fila en la Tabla 6 resume los resultados en las 10 instancias de cada conjunto. Las primeras seis filas corresponden a las instancias generadas con los parámetros de capacidad 0.2 y las últimas seis filas con 0.3.

		T1 Heurístico Previo					SO			
Set	***n***	***f(M)***	**Dev**	**Best**	**Time**		***f(M)***	**Dev**	**Best**	**Time**
GKD-b2	50	1057	6,4	0	0		1123	0,0	10	0
GKD-b2	150	1132	1,8	2	0		1152	0,0	9	0
GKD-c2	500	60	20,1	0	47		75	0,0	10	19
SOM-a2	50	36	12,0	6	0		41	0,0	10	0
MDG-b2	500	354	13,8	1	49		412	0,5	9	65
GKD-b3	50	925	4,9	2	0		970	0,0	9	0
GKD-b3	150	1046	0,2	7	0		1034	1,2	3	1
GKD-c3	500	51	21,9	0	47		65	0,0	10	41
SOM-a3	50	15	5,0	9	0		15	5,0	9	0
MDG-b3	500	86	21,6	2	49		111	1,9	8	45

Tabla 6. Comparación con el heurístico previo

La Tabla 6 muestra claramente la superioridad de nuestro método con respecto a la T1 heurística anterior (ver, por ejemplo, la columna Dev en ambos métodos). Esto es de esperar ya que T1 es un método de aproximación relativamente simple, con un rendimiento de garantía, y SO es una metaheurística compleja. Sin embargo, tengamos en cuenta que SO no solo supera a T1 en calidad de solución, sino que también es competitivo en tiempo de ejecución. Por otro lado, T1 presenta un muy buen rendimiento considerando su simplicidad, con resultados notables en los conjuntos de instancias GKD-b3 y SOM-a3 (7 y 9 mejores soluciones de cada 10 en cada conjunto, respectivamente). Ejecutamos SO para un tiempo de CPU similar al T1 para una comparación justa.

Para complementar el análisis anterior, comparamos ambos métodos con dos pruebas no paramétricas bien conocidas para comparaciones por pares: la prueba de **Wilcoxon y la prueba Signos**.

Los p-valores resultante de 0,00, en la prueba muestran que los valores comparados no provienen del mismo método.

Por otro lado, la prueba de Signos calcula el número de instancias en las que un algoritmo reemplaza a otro. Los p-valores resultantes de 0,00 indican nuevamente que hay un claro ganador entre ambos métodos (SO) cuando consideramos todas las instancias en el conjunto de referencia.

En el siguiente experimento, comparamos SO con las soluciones obtenidas con Cplex. No incluimos en este experimento las instancias de gran tamaño (n = 500), ya que Cplex no puede resolverlas en el límite de tiempo de 6,000 segundos considerado. La Tabla 7 muestra las soluciones de este experimento, donde el valor de desviación, Dev, se calcula con respecto al límite superior obtenido con Cplex. Por otro lado, el número de mejores soluciones, Best, considera el límite inferior obtenido con Cplex y la mejor solución obtenida con SO. Notemos que cuando Cplex termina la exploración por ramificación y acotamiento, ambos valores, inferior y superior, son los mismos.

		Cplex					**SO**			
Set	n	$f(M)$	**Dev**	**Best**	**Time**		$f(M)$	**Dev**	**Best**	**Time**
GKD-b2	50	112,3	0,0	10	2,8		112,3	0,0	10	0,1
GKD-b2	150	118,6	4,1	10	2926,0		118,4	4,3	6	37,5
SOM-a2	50	4,1	0,0	10	1,8		4,1	0,0	10	0,0
GKD-b3	50	97,8	0,0	10	4,3		97,8	0,0	10	1,6
GKD-b3	150	107,4	31,4	8	3600,0		107,2	31,6	5	56,3
SOM-a3	50	2,1	0,0	10	6,0		1,8	13,3	7	0,1

Tabla 7. Comparación con Método Exacto.

Los resultados en la Tabla 7 indican que Cplex es capaz de resolver las pequeñas instancias, con n = 50, y llegar a la optimalidad en tiempos de ejecución moderados (menos de 10 segundos). Sin embargo, cuando pasamos a las instancias de tamaño mediano, con n = 150, solo resuelve una fracción de ellas dentro del límite de tiempo de 3 600 segundos (veamos las desviaciones de 4,1 y 31,4 en los conjuntos GKD-b2 y GKD-b3 respectivamente). Como

se mencionó anteriormente, Cplex no pudo resolver las instancias de gran tamaño n = 500 en ninguno de los casos analizados dentro de los 3 600 segundos considerados.

Considerando ahora nuestro método SO, podemos ver en la Tabla 8 que en las pequeñas instancias de GKD (los conjuntos GKD-b2 y GKD-b3 con n = 50), es capaz de igualar las 20 soluciones óptimas en un tiempo de ejecución muy pequeño (menos de 2 segundos). En los casos más grandes en estos conjuntos, con n = 150, nuestra heurística de SO obtiene buenas soluciones, pero no es óptima en todos los casos. En particular, en GKD-b2 obtiene 6 de cada 10, y en GKD-b3, 5 de 10. Sin embargo, consideremos que solo lo ejecutamos durante un tiempo de ejecución moderado (menos de 1 minuto).

La Tabla 7 revela que las instancias en el conjunto SOM-a3 constituyen un desafío para los métodos heurísticos. Estas son instancias pequeñas con n = 50 en las que Cplex puede calcular las soluciones óptimas. Sin embargo, nuestro método solo obtiene 7 de las 10 soluciones óptimas. Teniendo en cuenta que en la Tabla 7, T1 muestra un rendimiento similar al SO en este conjunto, podemos concluir que estos casos son difíciles de resolver para estas dos heurísticas. Sin embargo, en todos los casos pequeños, incluido este conjunto, ejecutamos nuestro método durante un tiempo de CPU muy corto (0,1 segundos), en línea con los trabajos anteriores.

En el siguiente experimento, comparamos SO con el conocido LocalSolver. Ejecutamos este método con la formulación F2 que se muestra en el capítulo 1. Ejecutamos ambos métodos con un límite de tiempo de 60 segundos. La tabla 8 muestra los resultados.

		LocalSolver				SO			
Set	n	$f(M)$	**Dev**	**Best**	**Time**	$f(M)$	**Dev**	**Best**	**Time**
GKD-b2	50	1123,3	0,0	10	4	1123,3	0,0	10	0
GKD-b2	150	1181,0	0,3	7	34	1181,5	0,3	6	15
GKD-c2	500	58,2	21,9	0	47	74,6	0,0	10	19
SOM-a2	50	4,0	90,0	1	3	41,0	0,0	10	0
MDG-b2	500	56,6	83,5	0	59	343,5	0,0	10	25
GKD-b3	50	977,9	0.0	10	12	969,9	0,7	7	0
GKD-b3	150	1072,5	0,1	8	26	1068,5	0,6	4	47
GKD-c3	500	57,4	11,5	0	45	64,9	0,0	10	41
SOM-a3	50	0,0	100,0	0	1	15,0	25,0	5	0
MDG-b3	500	35,1	65,3	0	57	104,7	0,0	10	45

Tabla 8. Comparación del SO con una Heurística de propósito general.

La Tabla 8 muestra que, en términos generales, SO obtiene mejores soluciones que LocalSolver en tiempos de ejecución más cortos. Esto es de esperar ya que SO se personaliza para el problema de dispersión con capacidades, mientras que LocalSolver es un solucionador general, aunque vale la pena mencionar, que también implementa metaheurísticas complejas, y es hoy en día un método de referencia, dado su notable rendimiento en muchos problemas de optimización combinatoria. De hecho, en los conjuntos GKD-b2 (n = 50 y n = 150), LocalSolver y SO se comportan de manera muy similar; en los conjuntos de GKD-b3 (n = 50 y n = 150), LocalSolver se desempeña mejor que SO; pero en el resto de los conjuntos, SO claramente supera a LocalSolver. Para confirmar la superioridad de nuestro método, realizamos las dos pruebas estadísticas descritas anteriormente, **el Wilcoxon y los Signos**, y obtenemos en ambos casos un p-valor de 0,00, lo que indica que hay diferencias significativas entre ambos métodos.

Los resultados en la Tabla 8 también confirman que los conjuntos de instancias de SOM son un desafío para los métodos heurísticos y, por lo tanto, podemos concluir que todavía hay espacio para mejorar los desarrollos heurísticos para este problema.

Para complementar el análisis anterior, ejecutamos los tres métodos heurísticos en comparación durante un tiempo de ejecución relativamente largo (500 segundos) en una instancia grande representativa de MDG-b. La Figura 18 muestra los perfiles de búsqueda en los que se representa la mejor solución obtenida con cada método para ver su evolución a lo largo del tiempo.

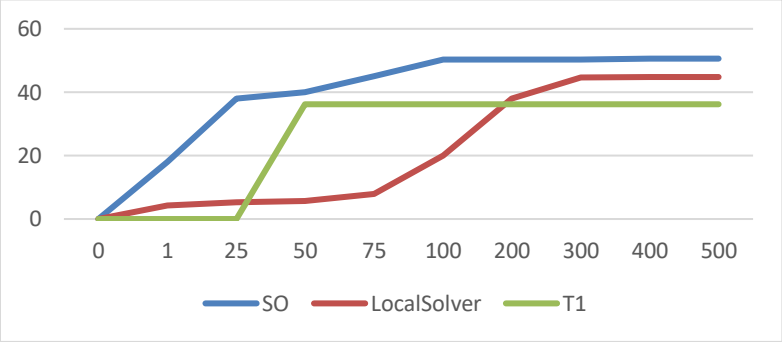

Figura 15. Perfiles de búsqueda en instancias en MDG-b

La Figura 18 muestra que SO es el método principal en todo el período de búsqueda. LocalSolver requiere un tiempo de configuración un poco más largo, y comienza a obtener soluciones de alta calidad después de 100 segundos de exploración. Finalmente, dado que T1 es un método simple, una vez que obtiene una solución, no hay ninguna mejora adicional. Todos los métodos se estancan después de 300 segundos, y por lo tanto no tiene sentido ejecutarlos más tiempo. La Figura 19 muestra un perfil de búsqueda similar para una instancia de GKD-c.

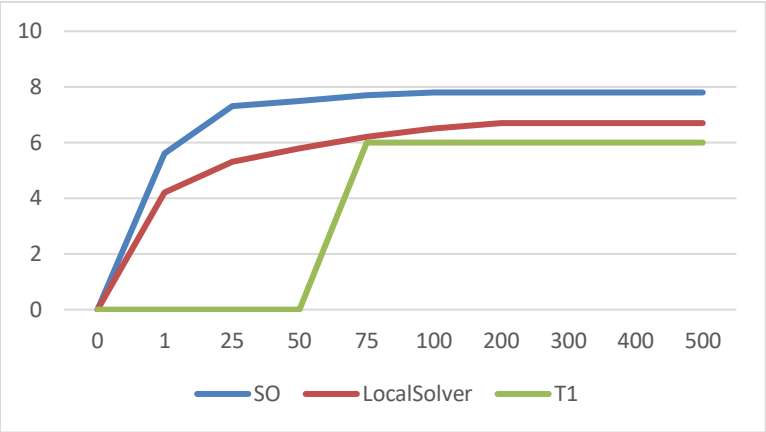

Figura 16 Perfiles de búsqueda en instancias GKD-b

4.5 Un caso de Aplicación para el problema de la dispersión.

4.5.1 Proyecto de Creación de Centros de Capacitación y Adiestramiento.

A manera de ilustración de como funciona el heurístico propuesto, y aprovechando la oportunidad para sugerir, a manera de iniciativa, la creación y ubicación de centros de capacitación y adiestramiento en una nueva comunidad, presentamos el siguiente ejemplo.

En primer lugar, nos ubicaremos en un contexto geográfico: La provincia de Colón, una de las 10 provincias de la República de Panamá. En su territorio se localiza la sección norte del canal de Panamá; es el principal puerto para el tráfico de casi toda la mercancía de importación y reexportación del país. En importancia, Colón es la segunda ciudad de la República.

Como una iniciativa del Gobierno de la República de Panamá para fortalecer el desarrollo social, el reordenamiento urbano y para mejorar la calidad de vida de más de 25,000 habitantes de la provincia de Colón, se crea un Proyecto de Renovación Urbana e Integración Humana, con el fin de promover una reingeniería habitacional y comercial en las avenidas principales de la provincia, para promocionar el Puerto Libre, la recuperación de edificios históricos y sitios de interés Nacional, además de beneficiar a los pobladores de la Provincia.

El proyecto de es un plan social que cuenta con 167 edificios con apartamentos de dos recámaras en un área cerrada de 48m² y de tres recámaras, con 57m², ubicado en la Provincia

de Colón, Corregimiento de Cristóbal, entre la Policlínica Hugo Spadafora Franco y el residencial Los Lagos (la Feria). Este proyecto integra áreas institucionales, una subestación de policía, cuartel de bomberos, corregiduría, áreas comerciales, infoplazas, centros educativos, atención médica, estaciones de bombeo, obras civiles, entre otras, que permitirán el desarrollo y crecimiento de las personas, como base de un desarrollo humano más incluyente y equitativo.

Cuando se crean proyectos de interés social, surgen iniciativas que ayudan al miembro de la comunidad en formación a mejorar su situación personal, profesional y económica. Una de esas iniciativas es la creación de centros de capacitación y adiestramiento para la formación de pequeñas empresas, centros que se ubicarán en la comunidad y cuyos facilitadores serán personas de la misma comunidad, lográndose plazas de empleos que contribuyan al engrandecimiento y mejora continua de la comunidad.

A continuación, exponemos un mapa de la ubicación del proyecto.

Figura 17. Mapa actual de la ubicación de la nueva barriada en Altos de Los Lagos, provincia de Colón.

Para el desarrollo del proyecto, se cuenta con 50 localidades donde se ubicarán los centros y las distancias medida entre cada punto. En la tabla siguiente se presenta la capacidad para cada centro.

localidad	capacidad	localidad	capacidad	localidad	capacidad	localidad	capacidad
1	210	14	150	27	120	39	120
2	120	15	120	28	120	40	250
3	210	16	210	29	150	41	250
4	120	17	150	30	210	42	120
5	120	18	180	31	210	43	120
6	210	19	210	32	120	44	120
7	180	20	120	33	150	45	210
8	150	21	120	34	180	46	250
9	180	22	150	35	180	47	180
10	150	23	150	36	120	48	180
11	210	24	150	37	210	49	210
12	120	25	180	38	250	50	250
13	180	26	180				

Tabla 9. Capacidad para cada una de las 50 localidades

La figura 18, muestra la ubicación de las 50 localidades.

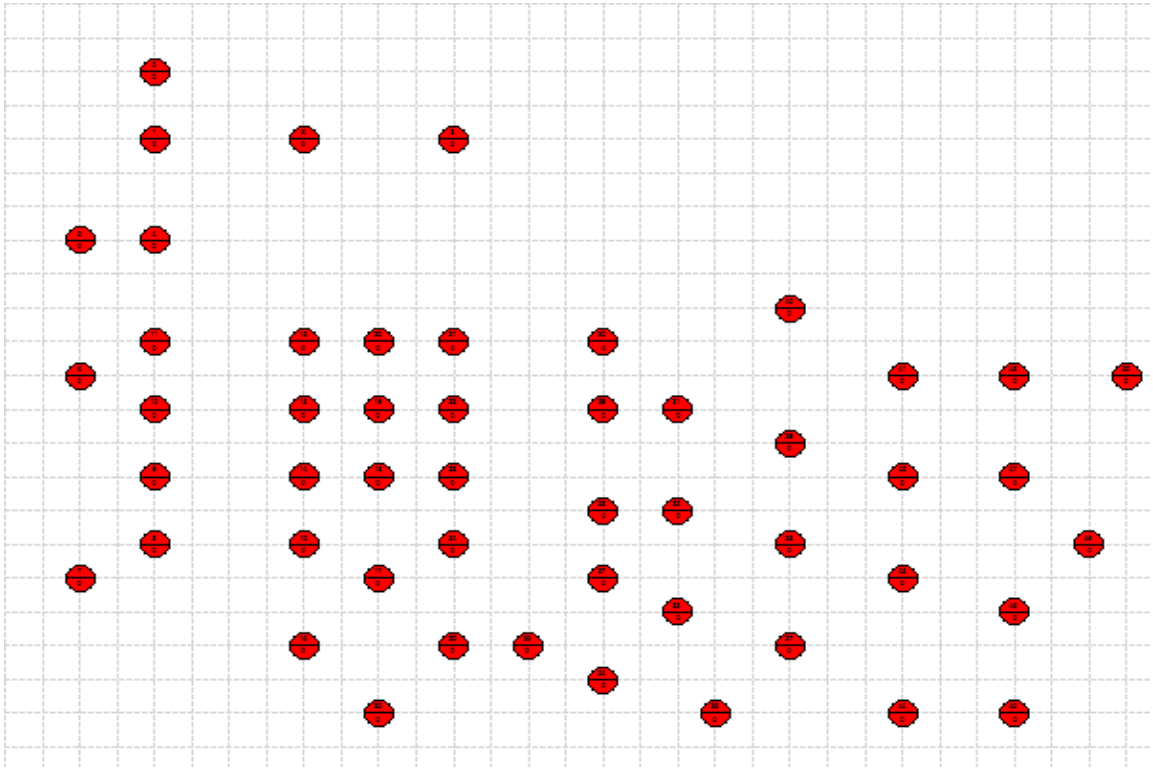

Figura 18. Gráfico que muestra la ubicación de cada localidad. [1]

[1] Este gráfico fue construido con el software Grafo versión 1.3.5

Al aplicar nuestra propuesta, y en menos de 2 segundos, obtenemos que la mejor ubicación para estos centros se da en nueve localidades,

$$M = \{16, 23, 24, 37, 38, 40, 44, 48, 49\}$$

y un valor de capacidad total de 1890 personas distribuidos en estos centros.

A continuación, en la figura siguiente se muestra en rojo la posición de las nueve localidades escogidas para ubicar los centros. En términos generales, se espera que los puntos estén más dispersos y mejor distribuidos, sin embargo, en nuestro problema no ocurre de esa manera y esto es debido a las condiciones de capacidad dadas en el problema.

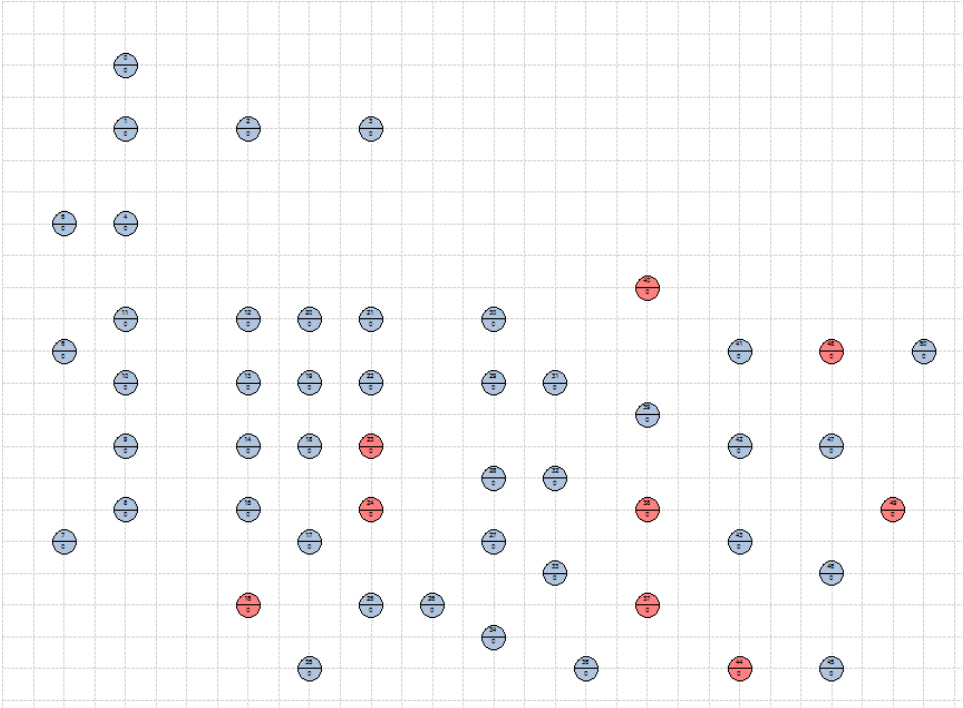

Figura 19. Diagrama donde se muestran la selecciónde los puntos escogidos por el algoritmo.

CONCLUSIONES

Mediante esta investigación hemos realizado un aporte significativo para la resolución del problema de localización de ubicaciones no deseables; un problema de Dispersión, que pertenece a los problemas de Optimización Combinatoria. En este sentido y de acuerdo con la literatura consultada, se han desarrollado diversos algoritmos relacionados con este problema de dispersión donde se considera principalmente la distancia entre instalaciones. En esta investigación de tipo experimental, hemos analizado este problema considerando además de las distancias, la capacidad asociada con cada ubicación.

En este documento se propone una heurística híbrida para el problema de la dispersión con capacidades. Esta heurística utiliza componentes de GRASP, búsqueda de vecindad variable (VNS) y oscilación estratégica (SO). Para encontrar una buena configuración para nuestra heurística consideramos cuatro procedimientos constructivos (que incluyen vecindarios constructivos y destructivos) con un parámetro para controlar la aleatorización en el proceso de selección y otro parámetro para establecer el nivel de oscilación en el SO. Nuestra extensa experimentación preliminar, reportada en las Tablas 2, 3, 4 y 5, establece valores apropiados para estos parámetros, proporcionando un buen balance entre las tres metodologías en términos de intensificación y diversificación de búsqueda.

Para este trabajo se tenían dos objetivos: experimentar con la hibridación de GRASP, VNS y SO y, en el proceso, desarrollar un procedimiento de vanguardia para el problema de la dispersión capacitada. Creemos que hemos logrado el primer objetivo con el diseño propuesto, simplemente llamado SO. El mérito de este método híbrido es que conserva las características principales de cada elemento al combinar sus capacidades. En términos de nuestro segundo objetivo, los resultados reportados en las Tablas 6, 7 y 8 son muy fuertes a favor de SO. En particular, las comparaciones con Cplex, Localsolver y un método específico anterior, T1, muestran claramente que SO puede obtener soluciones de alta calidad en tiempos de cómputo cortos.

REFERENCIAS

Alidaee, B. y. (2007). Solving the maximun edge weight clique problem via unconstrained quadratic programming. European Journal of Operational Research, 592-597.

Bosque S., J. y Franco, S. (1995). Modelo de Localización-Asignación y Evaluación Multicriterio para la Localización de Instalaciones No-Deseables. Serie Geográfica, No5, 97-112

Bosque, J., Gómez, M. y Palm, F. (2006) Un Nuevo Modelo para Localizar Instalaciones No Deseables: Ventajas Derivadas de la Integración de Modelos de Localización-Asignación y SIG. Cuadernos Geográficos, 53-68

Delgado, C. (2002). Nuevas Técnicas Metaheurísticas: Aplicación al Transporte Escolar. Universidad de Burgos. Servicio de Publicaciones. Colección: Estudio y Monografías. 71-73

Duarte A., Martí R. (2007) Tabu search and GRASP for the MDP. European Journal of Operational Research 178, 71-84.

Duarte, A. y. (2007). Tabu Search and GRASP for the Maximum Diversity Problem. European Journal of Operational Research, Vol 178, 71-84.

Duarte, A., Sánchez-Oro, J., Mladenovic, N., Todosijevic, R. (2018) Variable Neighborhood Descent, In: Martí, Resende, Pardalos (Eds) Handbook of Heuristics, Springer.

Erkut, E. (1990). The discrete p-dispersion problem. European Journal of Operational Research, 46(1), 46-60.

Erkut, E. Neuman, S. (1989). Analytic Models for locating undesirable facilities. European Journal of Operational Research, Vol. 40, 275-291.

Feo, T., Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109 133.

Gallego, M., Duarte, A., Laguna, M., and Martí, R. (2009) Hybrid heuristics for the maximum diversity problem. Computational Optimization and Applications, 44(3):411.

Ghosh, J. (1996). Computational aspects of the maximun diversity problem. Operations Research Letters, 175-181.

Glover F., K. C. (1993). Analyzing and Modeling the Maximun Diversity Problem by Zero-One programming. Decision Sciences 24(6), 1171-1185.

Glover F., Kuo C.C., Dhir K.S. (1998) Heuristic algorithms for the maximum diversity problem. Journal of Information and Optimization Sciences 19; 109–132.

Glover, F. K. (1998). Heuristic Algoriths for the Maximun Diversity Problem. Joournal of Information and Optimization Sciences, Vol 19, 109-132.

Glover, F., Laguna, M. (1997) Tabu search. Kluwer, Norwell, MA.

Glover, H. (1977). Selecting a sunset of maximum diversity. Boulder: ms/is report n°77-9. Technical report, University of Colorado.

Hansen, P. N. (2003). Búsqueda de Entornos Variables. Revista Iberoamericana de Inteligencia Artificial. No19, Vol.2, 77-92.

Martí, R., Duarte, A., and Gallego, M. (2019) MBPLIB- Maximum Diversity Problem Library, http://grafo.etsii.urjc.es/optsicom.

Martí, R., Gallego, M., and Duarte. A. (2010) A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research, 200(1):36–44.

Martí, R., Sandoya F. (2013) "GRASP and Path Relinking for the equitable dispersion problem," *Computers and Operations Research* 40, 3091-3099.

Melián, B, Moreno, J, Marcos, J. (2003), Metaheuristics: A Global View. Revista Iberoamericana de Inteligencia Artificial No. 19, pp. 7-28

Palubeckis, G. (2007) Iterated tabu search for the maximum diversity problem. Applied Mathematics and Computation, 189:371383.

Prokopyev O.A., Kong N., Martinez-Torres D.L. (2009) The equitable dispersion problem. European Journal of Operational Research 197, 59-67.

Prokopyev, O. K.-T. (2009). The equitable dispersion problem. European Journal of Operational Research, 59-67.

Resende M.G.C., Martí R., Gallego M., Duarte A. (2010) GRASP with path relinking for the max-min diversity problem, Computers and Operations Research 37, 498-508.

Rosenkrantz, D.J., Tayi, G.K., Ravi, S.S. (2000) Facility dispersion problems under capacity and cost constraints, Journal of Combinatorial Optimization 4, 7-33.

Sadiq, S. M. y Habib, Y. (1999). Iterative Computer Algorithms with Applications in Engineering. Solving Combinatorial Optimization Problems.Wiley. Editor IEEE Computer Society Press, 99/9/1

Sandoya, F. Un modelo para el problema de la diversidad máxima. Un estudio para la selección de lo mejor y lo más diverso. PhD tesis, Ingeniería de Sistemas - Investigación de Operaciones. Universidad Nacional Autónoma de México, Dec. 2013.

Sandoya, F., Martínez-Gavara, A., Aceves, R., Duarte, A., Martí, R. (2018) Diversity and Equity Models, In: Martí, Resende, Pardalos (Eds.) Handbook of Heuristics, 979-998 Springer.

Silva G.C., Ochi L.S., Martins S.L. (2004) Experimental Comparison of Greedy Randomized Adaptive Search Procedures for the Maximum Diversity Problem. In: Ribeiro C, Martins C Simone L. (Eds.), Experimental and efficient algorithms, vol. 3059. Springer: Berlin. 498–512.

Silva, G.C., Andrade, M.R.Q., Ochi, L.S., Martins, S.L., Plastino, A. (2007) New Heuristics for the Maximum Diversity Problem, Journal of Heuristics 13(4), 315-336.

Wang, J. y. (2009). Competitive Hopfield Network Combined with Estimation of Distribution for Maximun Diversity Problems. IEEE Transactions on system, man and cybernetics, Vol. 39, 1048-1065.

Páginas de Internet

Hernández, Ana. (s.f.) Programación Lineal Entera. Recuperado de https://www.monografias.com/trabajos66/programacion-lineal-entera/programacion-lineal-entera2.shtml

GEO Tutoriales (10/8/2011). Diferencias entre la Programación No Lineal y la Programación lineal. Recuperado de https://www.gestiondeoperaciones.net/programacion-no-lineal/diferencias-entre-la-programacion-no-lineal-y-la-programacion-lineal/

Merino M., María. (s.f.) Técnicas Clásicas de Optimización. Parte I: Programación Lineal y No Lineal Facultad de Ciencia y Tecnología, Departamento de Matemática Aplicada y Estadística e Investigación Operativa UPV/EHU. Recuperado de http://www.ehu.eus/mae/html/prof/Maria_archivos/plnlapuntes.pdf

Wikipedia. (13/10/19). Programación Lineal. Recuperado de https://es.wikipedia.org/wiki/Programaci%C3%B3n_lineal#:~:targetText=La%20programaci%C3%B3n%20lineal%20es%20el,ecuaciones%20o%20inecuaciones%20tambi%C3%A9n%20lineales.

Lynxblack. (13/05/2015). La teoría de la complejidad computacional en palabras (no tan) complejas. Recuperado de https://frikosfera.wordpress.com/2015/03/13/la-teoria-de-la-complejidad-computacional-en-palabras-no-tan-complejas/

Moyano, Nelson. (4/03/2018). ¿Qué es la Complejidad Computacional? Recuperado de https://medium.com/@nelramoyano/qu%C3%A9-es-la-complejidad-computacional-3a556e557973

http://www.cemosa.es/referencia/proyecto-habitacional-alto-los-lagos-colon-panama/

Anexo.

Conceptos Relacionados

Algoritmo Heurístico: Un algoritmo heurístico es un procedimiento que permite encontrar una solución y que suelen diseñarse de modo específico para cada problema. En Programación Matemática heurística suele hacer referencia a un procedimiento que busca una solución, aunque no garantiza encontrar la mejor solución. En Inteligencia Artificial se suele denominar función heurística a aquella que dirige la búsqueda (o construcción) de una solución, utilizando algún mecanismo más o menos sencillo.

Un buen algoritmo heurístico debe ser eficiente, bueno y robusto. Esto es debe requerir un esfuerzo computacional realista, su resultado debe estar suficientemente cerca del óptimo, y la probabilidad de obtener una mala solución debe ser baja.

Optimización combinatoria: es una rama de la optimización de las matemáticas aplicadas fuertemente relacionada con la investigación operativa, la teoría algorítmica y la teoría de la complejidad computacional. Los algoritmos de optimización combinatoria resuelven problemas que se creen difíciles en general, por la vía de explorar el, habitualmente grande, espacio de soluciones del citado problema. Los buenos algoritmos de optimización combinatoria lo logran por la vía de reducir el tamaño efectivo del espacio a buscar y explorando de modo eficiente dicho espacio.

Prueba de los rangos con signo de Wilcoxon es una prueba no paramétrica para comparar el rango medio de dos muestras relacionadas y determinar si existen diferencias entre ellas. Se utiliza como alternativa a la prueba t de Student cuando no se puede suponer la normalidad de dichas muestras. Debe su nombre a Frank Wilcoxon, que la publicó en 1945.

Programación Lineal: La Programación Lineal también conocida como optimización lineal, trata sobre la optimización (maximizar o minimizar) de una función lineal, llamada función objetivo, sobre un poliedro convexo definido por un conjunto de restricciones lineales no negativas que pueden ser expresadas mediante sistemas de ecuaciones o inecuaciones lineales.

Programación Lineal Entera: La programación lineal entera es el conjunto de problemas de programación lineal para los cuales todas o parte de sus variables pertenecen a los números enteros. Generalmente estas variables toman los valores 0 o 1 (variables binarias) y se usan para representar condiciones lógicas y analizar situaciones complejas.

Programación No Lineal: Los problemas de Programación No Lineal se caracterizan por tener relaciones no lineales; tanto la función objetivo del problema que se desea optimizar, así como las restricciones corresponden a ecuaciones cuyas variables tienen exponente distinto de uno.

Programación Cuadrática: Existen diversos tipos de problemas de acuerdo con las características de la función objetivo de PNL; uno de estos tipos son los relacionados con la Programación Cuadrática cuya función objetivo está determinada por el cuadrado de una variable o el producto de dos variables sujeta a restricciones lineales de igualdad o desigualdad.

Complejidad de los Problemas de Optimización Combinatoria.

La teoría de la complejidad trata de discernir entre lo problemas de optimización que son fáciles de resolver y los difíciles que suponen hoy día un reto para las metodologías existentes. En este sentido, la clase P, incluiría a los fáciles, para los que existe un método de resolución cuyo tiempo de cómputo es moderado, lo cual se cuantifica acotándolo por una expresión matemática polinómica respecto al tamaño del problema. La investigación en optimización se centra actualmente en los problemas que no pertenecen a esta clase, los llamados NP, y que por lo tanto son considerados difíciles de resolver. El problema abordado en esta tesis doctoral pertenece a esta última clase.

CPLEX es un solucionador comercial de alto rendimiento flexible para programación lineal (LP) y programación entera mixta (MIP). También puede usarse para la resolución de programación cuadrática. Es capaz de resolver problemas de optimización reales con muchas variables, incluye un algoritmo paralelo distribuido para la programación de enteros que permite aprovechar múltiples sistemas para resolver prolemas difíciles. En la mayoría de los casos, la configuración por defecto es suficiente para resolver problemas con tiempos de ejecución excelentes. El algoritmo de ramificación y acotación de Cplex para la resolución de problemas MIP usa características avanzadas como planos de corte o heurísticas para encontrar soluciones enteras.

LocalSolver es un sistema de optimización comercial que proporciona un rendimiento extremadamente sólido en múltiples clases de problemas de optimización. La búsqueda se inicia desde una solución generada por un procedimiento básico aleatorio codicioso. Esta se realiza en la región factible y los movimientos se realizan para transformar una solución en otra. Como su nombre lo indica, LocalSolver intenta encontrar óptimos locales mediante

técnicas de ascenso estándar (para problemas de maximización). Las heurísticas integradas permiten que el proceso seleccione movimientos que no mejoran para escapar de la optimalidad local. Estas heurísticas incluyen modelos probabilísticos como los típicos de la metodología de recocido simulados. Un gran conjunto de movimientos está disponible durante la búsqueda y la selección de los movimientos a intentar, se ajusta dinámicamente. LocalSolver es gratuito para usos académicos y se puede descargar desde http://www.localsolver.com/.